Journal of Artificial Intelligence Research 77 (2023) 971-1019 Submitted 09/2022; published 07/2023

Information Lattice Learning

Haizi Yu HAIZIYU@UCHICAGO.EDU
James A. Evans JEVANS@UCHICAGO.EDU
Knowledge Lab, University of Chicago,
1155 E 60th Street, Chicago, IL 60637 USA

Lav R. Varshney VARSHNEY@ILLINOIS.EDU
Coordinated Science Lab, University of Illinois at Urbana-Champaign,
1308 W Main Street, Urbana, IL 61801 USA

Abstract

We propose Information Lattice Learning (ILL) as a general framework to learn rules of a signal (e.g., an image or a probability distribution). In our definition, a rule is a coarsened signal used to help us gain one interpretable insight about the original signal. To make full sense of what might govern the signal's intrinsic structure, we seek multiple disentangled rules arranged in a hierarchy, called a *lattice*. Compared to representation/rule-learning models optimized for a specific task (e.g., classification), ILL focuses on *explainability*: it is designed to mimic human experiential learning and discover rules akin to those humans can distill and comprehend. This paper details the math and algorithms of ILL, and illustrates how it addresses the fundamental question "*what* makes X an X" by creating rule-based explanations designed to help humans understand. Our focus is on explaining X rather than (re)generating it. We present applications in knowledge discovery, using ILL to distill music theory from scores and chemical laws from molecules and further revealing connections between them. We show ILL's efficacy and interpretability on benchmarks and assessments, as well as a demonstration of ILL-enhanced classifiers achieving human-level digit recognition using only one or a few MNIST training examples (1–10 per class).

1. Introduction

With recent progress in AI, there is increasing need for general (Goertzel & Pennachin, 2007) and explainable (Adadi & Berrada, 2018; Molnar, 2019) AI, which exhibit broad, human-like cognitive capacities. One common pursuit is to move away from "black boxes" designed for specific tasks toward achieving generalization via abstractions made from only a few examples. Such a generalization process should require neither unlimited priors nor unlimited data, taking only "primitive priors" and "small data" as inputs (Chollet, 2019). In this pursuit, we present a new, task-nonspecific framework, called *Information Lattice Learning (ILL)*. The goal is to learn rules from data, similar to human experiential learning, e.g., distilling music theory from music, in a human-interpretable form similar to a textbook (as shown below).

The term *information lattice* was first coined by C. E. Shannon (1953), but remains largely conceptual and unexplored. In the general context of abstraction/representation learning, we develop lattices that coincide with Shannon's. We also generalize the original definition: an information lattice here is a hierarchy of abstracted representations (rule lattice) and we bring learning into the lattice (rule learning). This yields the name *information lattice learning*.

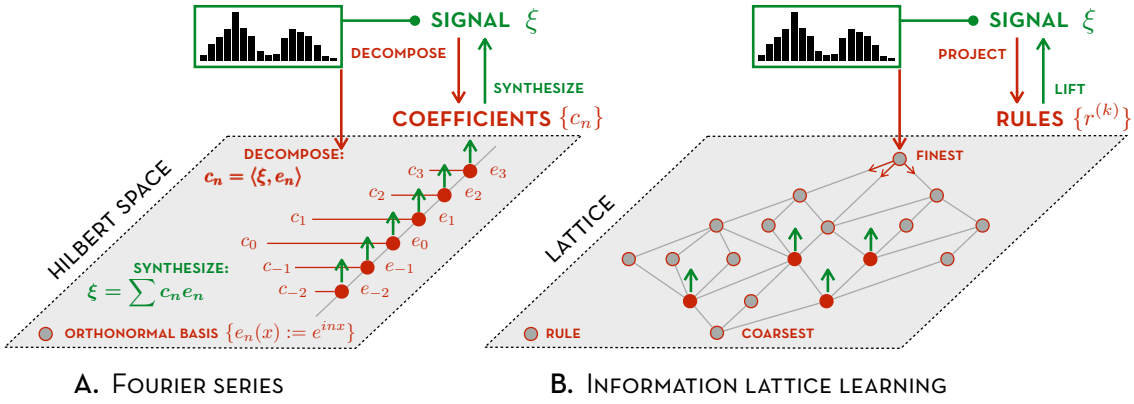


Figure 1: ILL and Fourier series: decompose a signal into simple components (rules).

Fundamental question. ILL aims to solve "what makes X an X" or "what makes X differ from Y", emphasizing the *what—explaining* X rather than generating X or predicting labels X,Y. ILL explains a signal (e.g., an image or a probability distribution) to people via human-like and human-interpretable abstractions or coarsenings of that signal called *rules*. As a signal may be viewed from several angles, one may aspire for several rules to collectively explain a large part of the signal, with each explaining a unique aspect. Solving this question enables rule-based explanations designed to help humans understand and learn complex sensory signals. Note that a music AI classifying concertos, or generating concertos that mimic the masters, does not necessarily produce human insight about what makes a concerto a concerto or the best rules a novice composer might employ to write one. The rules we seek may be used in classification later, but they are primarily built for understanding. Instead of optimizing a task-specific objective (e.g., classification error), ILL balances among objectives favoring *fewer*, *simpler* rules for interpretability, as well as more *essential* rules for effectiveness, as we formalize below.

Central intuition. The mathematical intuition behind ILL is: *to break the whole into simple pieces*, similar to breaking a signal into Fourier series (Figure 1). Whereas Fourier analysis decomposes a signal in a Hilbert space via inner product (i.e., projection to orthonormal basis) and synthesizes it via weighted sum, ILL decomposes a signal in a hierarchical space called a *lattice*. Our goal is to restore human-like, hierarchical rule abstraction-and-realization via signal decomposition-and-synthesis in a lattice, called *projection-and-lifting*.

Model overview. ILL comprises two phases: I) lattice construction; II) learning/searching in the lattice (Figure 2A). This is similar to machine learning (ML) models comprising I) function class construction then II) learning in the function class. For example, we first construct a neural network architecture then learn optimal network parameters via back-propagation. ILL's construction phase is *prior-efficient*: it builds in universal priors consistent with human innate cognition—the Core Knowledge priors (Spelke & Kinzler, 2007)—then grows a lattice of abstractions. ILL's learning phase is *data-efficient*: it learns from "small data" as a signal, but searches for rich explanations of the signal via rule learning. Abstraction here is key to "making small data large". Notably, the construction phase is prior-driven, *not* data-driven. Data comes in only at the learning phase. As a result, the same lattice construction may be

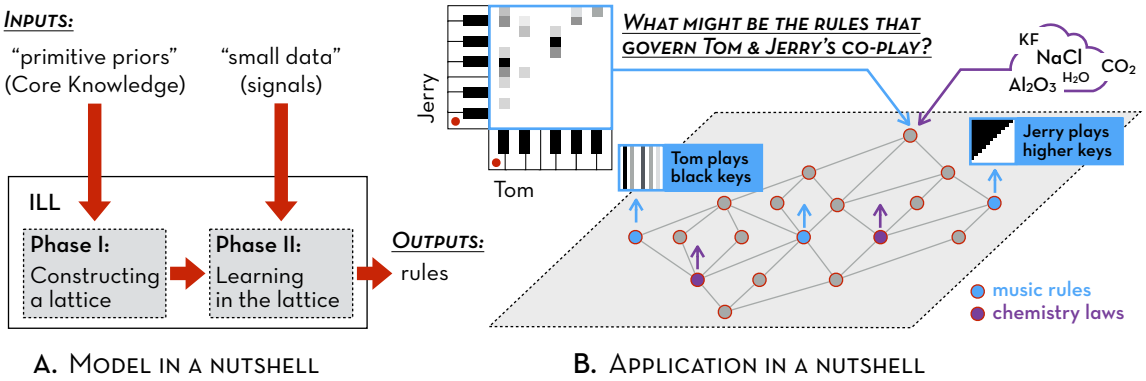


Figure 2: ILL comprises two phases: I) lattice construction; II) learning in the lattice. A lattice is first constructed regardless of the signal (prior-driven), then the lattice may be reused to learn rules (data-driven) of signals from different topics, e.g., music and chemistry.

reused or transferred from the learning phase on one problem to other data or even different subject domains like music and chemistry (Figure 2B). Featuring these two phases, ILL is thus a hybrid model that threads the needle between a data- and prior-driven model to reflect the idea of "starting like a baby and learning like a child" (Hutson, 2018).

Application overview. Imagine Tom and Jerry are playing two 12-key pianos simultaneously, one note at a time (Figure 2B). The frequency of two-note chords played gives a 2D signal plotted as a 12×12 grayscale heatmap. Inspecting this heatmap, what might underlie rules that govern their co-play? Is it the case that Jerry always plays higher keys while Tom always plays black keys? All grey pixels have a larger "Jerry-coordinate" and project to a black key along the "Tom-axis". Yet, rules like these in real contexts may not be obvious for human cognition to unravel. This paper includes learning music theory from scores and chemical laws from molecules as typical ILL applications. In addition, we show how ILL's common priors facilitate mutual interpretation of knowledge learned from the two disciplines. We also include ILL benchmarks and assessments, as well as a brief demonstration of ILL-enhanced simple classifiers achieving human-level digit recognition using only one or a few MNIST training examples.

2. Why Unsupervised Learning Baselines Are Unsatisfactory

Aiming for *human explainability*, our fundamental question "what makes X an X" differs from classic AI problems like classification or data reconstruction. Existing ML methods are unsatisfactory for this aim. Consider Tom and Jerry's heatmap in Figure 2B. Per Core Knowledge (Spelke & Kinzler, 2007) and Gestalt theory (Marr, 1982), two patterns are obvious to humans: Jerry uses higher keys; Tom uses black keys. In non-musical language, data are "above the diagonal" and form five "columns" (the black H_is atop Figure 3). Are these human-discernible concepts (clusters) also discernible to AI?

Considering the general goal of making sense of sensory inputs and forming concepts, unsupervised learning baselines like data clustering and dimensionality reduction are common first choices (Eisen et al., 1998; Zhou et al., 2018; Clark et al., 2019). Yet, both remain far

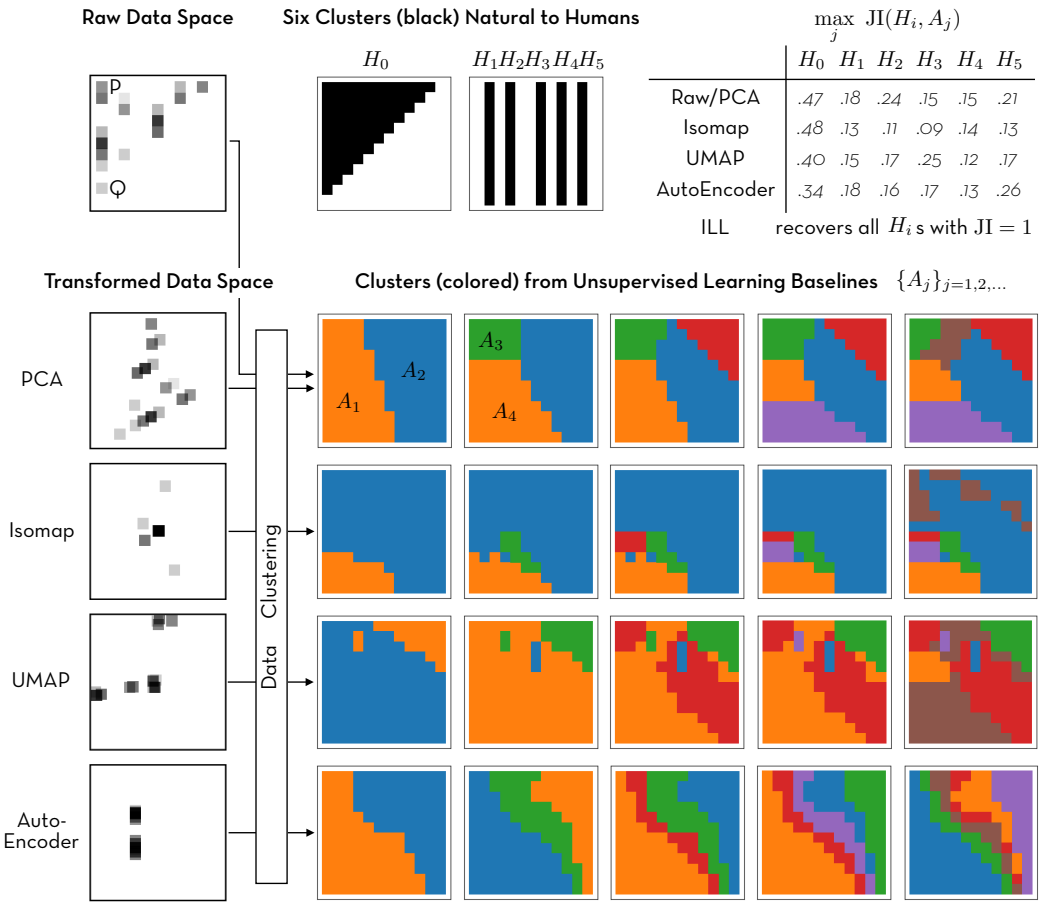


Figure 3: Unsupervised learning baselines on the "Tom-and-Jerry" example from Figure 2B. Raw data are easy for humans to see two patterns: "above the diagonal" (H_0); five "columns" ($H_1, \ldots, H_5$). Running k-means on both raw and transformed (via manifold learning) spaces yields various clusters ($A_1, A_2, \ldots$). The Jaccard index for each pair $\mathrm{JI}(H_i, A_j)$ reveals how algorithm-generated clusters coincide with human-perceived ones, with $\mathrm{JI} = 1$ denoting an exact recovery. All listed unsupervised learning baselines are not able to recover any H_i to 50%. We will show that ILL can not only recover clusters with $\mathrm{JI} = 1$ on this example, but discover domain knowledge—requiring more than just clustering—in real cases.

from humans' intuitive perception and conceptualization. We show this in Figure 3 and use the Jaccard index (JI) to measure how human-discernible concepts may be recovered by the baselines. Specifically, for any human-discernible concept H_i (as a set) and any algorithm-produced cluster A_j, we use $\mathrm{JI}(H_i, A_j) := |H_i \cap A_j|/|H_i \cup A_j|$ to measure to what extent A_j recovers H_i, with $\mathrm{JI} = 1$ denoting an exact recovery.

Figure 3 shows that common baseline algorithms did not recover any human concepts ($\mathrm{JI} < 0.5$), suggesting they remain far from human conceptualization. This is regardless of whether data clustering is applied to raw or transformed data via manifold learning commonly used for dimensionality reduction (PCA, Isomap, UMAP, Autoencoder). We first observe four gaps below and then stress the most important limitation from the baselines.

(a) Heuristics like dimensionality or sparsity are too crude to approximate human-level abstraction. In Figure 3, data is already of small size and two dimensional with no computational burden or curse of dimensionality. Patterns apparent to humans are not for these baselines.

(b) Data clustering focuses on clustering present data, but not absent data or the structure of the entire data space. While all clustering methods in Figure 3 can trivially produce a single cluster including all present data, they fail to explain what is special about this single cluster, e.g., pointing out that raw data are all above the diagonal.

(c) Data clustering is tied to the metric space (raw or transformed) where data reside, making it unlikely for distant data to be in the same cluster. Without innate, human-like priors, it is hard to flexibly yield clusters with disconnected parts linked via symmetry or global shape. In Figure 3, despite being in the same "column", most methods (except for very coarse divisions) separate points P,Q based on their distance.

(d) Restoration-centric compression or dimensionality reduction ignores human comprehension. In Figure 3, the first principal component ($0.48 \times$ Jerry's key $+ 0.87 \times$ Tom's key) explains 80% of the data variance, but what does this linear combination mean? Is this a new synthesized music pitch? While data are condensed into five clear clusters in manifold learned by Isomap/UMAP, how should one interpret this manifold or post hoc interpret the five clusters? Even being able to interpret clusters post hoc from known knowledge is unsatisfactory. When new findings revealed from previously unseen clusters in a manifold, how to understand/evaluate new knowledge in the absence of suggested references requires interpreting the manifold itself.

Most importantly, knowledge discovery is not just compression and clustering. Even when clusters are clearly separated, explaining them is a distinct, nontrivial task. A clear separation may still be elusive, being possibly the entanglement of simpler clusterings. In Figure 3, one may obtain a single clustering (the *join*) by combining the two human-perceived clusterings, which is more elegant but less intelligible than decomposed ones. So, explaining data patterns is not about obtaining a single "clear cut" but inferring a *decomposition* into simpler clusterings. The lattice structure of partitions allows encoding decompositions that orthogonally augment hierarchical clustering—seeking "truly distinct" (i.e., incomparable) clusterings rather than unidirectional coarsening or refinement. Moreover, we bring learning into lattices to enable effective inference and reasoning via clustering decomposition. For instance, decomposing raw data in Figure 3 into two human-perceived clusterings effectively expresses the logic clause $H_0 \wedge (H_1 \vee H_2 \vee H_3 \vee H_4 \vee H_5)$. Further, human-distilled rules are not only about abstracting out clusters (as concepts) but describing abstract patterns, e.g., not only grouping mammals, birds, and fish into their own clusters but further explaining that "most birds fly". In Figure 3, it is clear that Tom always plays black keys, but is he also predominantly playing two of them (may be C# or F#)? In this paper, we also define a new, explicit, and stochastic form of "rules" beyond manifolds and clusters.

Common unsupervised learning baselines are incapable of solving our fundamental question. As such, a new learning paradigm is required that mimics human abstraction and materialization and that aims to offer people a panoramic explanation of data. By the end of this paper, we will show that ILL can recover all of the six human concepts (JI $= 1$) as well as many other rules in real music examples and other topic domains.

3. Related Intuition and Related Work

ILL incorporates both conceptual and algorithmic ideas from models in ML and signal processing. We also draw intuitions and algorithms from other areas including lattice theory, group theory, information theory, and optimization. We position ILL among several lines of existing research, explaining similarities and differences and why none alone is sufficient to solve our fundamental question in this paper.

Occam's Razor and data compression. ILL's core philosophy is *simplicity*, which is key to succinct insight and interpretation. In this way, ILL may be viewed as a special class of *data compression*. From this view, ILL's high-level architecture is similar to dimensionality-reduction models like PCA, autoencoders (Kramer, 1991), as well as models from algorithmic information theory (Chaitin, 1987; Chater & Vitányi, 2003) and compressed sensing (Eldar & Kutyniok, 2012). A key distinction among these "compression" models lies in the definition of simplicity. Whereas criteria like norms, sparsity, and description length are common, ILL expresses simplicity from its unique perspective on human explainability. When there is trade-off, simplicity is prioritized over reconstruction error. This can come at the cost of sacrificing fidelity but may be desired to maximize human learning, understanding, and creative thinking. Consider new music in a given style characterized by a probability distribution over style-consistent data. Creating rules—interpretable high-level principles—of the distribution not only enables discrimination and generation, but yields new data distributions under criteria like maximum entropy (MaxEnt) (Jaynes, 1957; Good, 1963) or even maximum deviation from the training distribution to encourage diversity and creativity.

Representation learning and casuability-based interpretability. Our focus on interpretability represents a shift from many representation learning models (Bengio et al., 2013) that aim to find the best representation for a specific task (e.g., classification, reconstruction) rather than for explanation. Interpretability in this paper is manifest not only in end results (i.e., the learned rules) but also model transparency (i.e., the rule-learning process). This makes our ILL framework *in-model interpretable*, which differs from black-box frameworks such as many deep neural networks that require post-hoc interpretation. Let us explicitly define and characterize our considered notion of interpretability.

- *Definition of interpretability (causability).* We adopt the notion of causability from the XAI community (Holzinger et al., 2020, 2021) as the formal definition of interpretability in this paper. Extending the notion of explainability that highlights technically decision-relevant parts of machine representations and machine models, *causability* measures the extent to which an explanation reaches a certain level of causal understanding for a human expert. Following Selbst and Barocas (2018), we specifically want causability in the sense of models that are neither inscrutable, where direct inspection may defy understanding, nor non-intuitive, resting on statistical relationships that defy human intuition.

- *Type of interpretability (in-model).* In-model interpretability is also known as intrinsic interpretability as opposed to post-hoc interpretability (cf. taxonomy of interpretability in XAI (Carvalho et al., 2019)). In-model interpretability implies model transparency, manifest at three different levels: simulatability, decomposability, and algorithmic transparency (Lipton, 2018). ILL is by design intrinsically interpretable. It simulates human

learning, has a decompositional nature, and is derived from sound theory (Sections 4 and 5). Our human assessment further provides empirical evidence (Section 7.2).

- *Interpretable to whom (domain experts).* A key question regarding interpretability is to clarify "interpretable to whom"—often another difference between intrinsic and post-hoc interpretability. In our case, we aim for intrinsic interpretability for those with domain expertise rather than for everyone. Domain experts may not have prior exposure to the AI system, but they are expected to use relevant domain expertise to understand AI outputs, often via some analytic processes.

  Being interpretable by domain experts differs from being obvious to everyone. Consider the representation of subatomic particle interactions through Feynman diagrams, which are cognitive tools initially used to make quantum chromodynamics computations easier, but can be applied in other fields such as solid-state theory. It requires some expertise in theoretical physics to understand what is being represented by the lines and squiggles, and very few people other than Feynman himself understood them when they were first developed (Kaiser, 2005). When we evaluate interpretability later in this paper, we consider interpretability by entry-level domain experts, namely students from a related subject area. As such, we can assess whether just entry-level expertise is sufficient to understand the representation of machine-generated knowledge, even though the knowledge is new or in a form that is previously unknown.

Causal inference and graphical models. Regarding inherently interpretable models, ILL is closely related to many self-explanatory, graph-based models, e.g., decision trees (Safavian & Landgrebe, 1991), Bayesian models (Koller & Friedman, 2009), and logic programming (Evans et al., 2021). Compared to tree-based models, a lattice (a directed acyclic graph) is structurally more general and avoids the downside of greedy algorithms typically used to grow a tree. Compared to probabilistic inference in Bayesian networks or reasoning in formal logic, edges in an information lattice are used to model partial ordering among human-like abstractions rather than conditional dependency or logical implications. Instead of using probabilistic or logical inference as driving mechanisms, learning in an information lattice mimics humans' bilateral processes of *abstraction* (distilling rules) and *realization* (applying rules). The two are respectively formalized by the *projection* and *lifting* operators—the twin inference engines beneath lattice learning, one the inverse of the other. As such, causal inference in ILL is in the form of rules and abstractions rather than statistical correlations, and ILL's abstraction-based stochastic rules differ in definition from those in formal logic.

Clustering. We define a rule based on the notion of *abstraction*, formalized by a partition or clustering of the data space. Abstraction in the form of clustering is the building block of ILL's Phase I for constructing a lattice. Our clustering model is closer to conceptual clustering (Michalski & Stepp, 1983; Fisher, 1987) than data clustering like k-means, and our clusters are formed in a prior-/mechanism-driven fashion. This reiterates how data is not required for ILL's construction phase, and the decomposed abstractions or clusterings may be reused later for multiple data sets or domains in the learning phase. Our clustering process is causal, directly explained by the priors and mechanisms used to combine priors, generate partitions, and form lattices. The more general notion of a partition lattice includes

hierarchical clustering as a special case, where the latter represents a linear lattice (called a chain) depicting only unidirectional coarsening or refinement.

Feature selection. In light of rule extraction, distilling the most informative rules from a lattice is similar to filtering out the most relevant features from a candidate set. There are three advantages to work with partitions and partition-defined rules over features, namely being more general, domain-agnostic, and structural. First, compared to a feature function, a *partition* captures the nature of human abstraction as an *equivalence relation* (Livingston, 1998), whereas naming equivalence classes is inessential (e.g., partitioning persons into two gender groups is enough to abstract the concept of binary gender; further labeling the two "M" and "F" is unnecessary). Any feature function induces a partition via preimage, but there are other fundamentally different ways, such as induced by symmetry or partial order (Chater & Vitányi, 2003). Second, we do not handcraft features with domain knowledge as an inclusive candidate set for feature selection. When using feature-induced partitions, all features are assembled from "primitive priors", e.g., simple arithmetic operators from Core Knowledge that are part of human innate cognition and domain-agnostic (Chollet, 2019; Spelke & Kinzler, 2007). Third, an information lattice has more structure than a set of candidate features. This structure is key, not only for spawning more partitions, but also for structured search—the heart of lattice learning—which fundamentally differs from gradient-based search for parametric models. As such, ILL enables discovery of new rules and new structures that can complete and extend existing domain knowledge.

Concept lattice. The model most related to ILL is *concept lattice* used in formal concept analysis (FCA) (Ganter & Wille, 2012; Ganter et al., 2016; Priss, 2006). Both use lattices and represent human-interpretable concepts consistently. In our case, a *concept* is an element in an abstraction defined by a cell in a partition, i.e., an equivalence class under an equivalence relation (Section 4). In FCA, a *formal concept* is defined as a set of objects (extent) sharing attributes (intent), i.e., objects equivalent under observed attributes. Our definition of a concept generalizes that of a formal concept in two ways. First, our partition is not induced from domain-specific attributes through formal logic and formal ontology, but from universal priors drawn from Core Knowledge (Section 5.1). Second, specifying a partition considers *all* of its cells as concepts (including knowns and unknowns), whereas specifying a set of formal concepts only considers those with respect to a given *formal context* (including known concepts only). In this way, our partition lattices generalize concept lattices in FCA, and are not generated or constrained by domain knowledge as those encoded in formal ontologies. Accordingly, ILL does not demand, but rather discovers, domain knowledge. In Appendix A, we detail a mathematical comparison between information and concept lattices, and exemplify the two lattices and their connection in biological taxonomy and music theory.

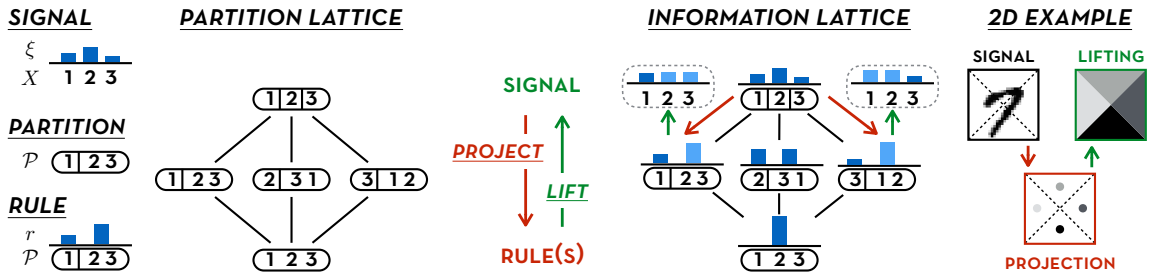


Figure 4: Basic information-lattice terms: signal, partition, rule, lattice, projection & lifting.

4. Information Lattice: Abstractions and Rules of a Signal

We introduce the basic terms necessary to define an ILL problem. The idea is to formalize human-like abstractions and abstraction hierarchies as partitions and partition lattices—the basis for defining rules that agree with our common sense and resemble human-distilled ones. Our formulation enables a computational approach for human-like rule abstraction and realization via the *projection* and *lifting* operators notated by $\downarrow$ and $\uparrow$ throughout the paper. Figure 4 sketches a pictorial illustration of all terms with examples.

Signal. A *signal* is generally defined by a function $\xi : X \to \mathbb{R}$, and we denote the set of all signals on a given domain X by $\mathcal{S}_X$. In this paper, we assume $\xi \geq 0$ and $X \subseteq \mathbb{R}^n$ is finite; however, this is not a limitation (cf. Appendix B). For example, we can use a signal $\xi : \{0, \ldots, 27\}^2 \to \mathbb{R}$ to represent a 28×28 grayscale image. A signal can also represent any probability distribution, e.g., the following $\xi : \{1, \ldots, 6\} \to \mathbb{R}$ representing the probability mass function (pmf) of rolling a biased die:

$$\xi(1) = 0.1,\ \xi(2) = 0.2,\ \xi(3) = 0.3,\ \xi(4) = 0.1,\ \xi(5) = 0.2,\ \xi(6) = 0.1 \tag{1}$$

Partition / abstraction. We use a partition $\mathcal{P}$ of a set X to denote an *abstraction* of X; we call a cell $C \in \mathcal{P}$ an (abstracted) *concept*. The intuition is simple: a partition of a set renders a "coarse-grained view" of the set, or more precisely, an equivalence relation on the set. In this view, we identify equivalence classes of elements (concepts) instead of individual elements. For example, the following abstractions of the six outcomes of a die roll

$$\{\{1,2\},\{3,4\},\{5,6\}\},\quad \{\{1,2,3,4\},\{5,6\}\},\quad \{\{1,3,5\},\{2,4,6\}\} \tag{2}$$

are partitions of the set $X = \{1, 2, 3, 4, 5, 6\}$.

Rule / representation. A *rule of a signal* $\xi : X \to \mathbb{R}$ is a "coarsened" signal $r_\xi : \mathcal{P} \to \mathbb{R}$ on a partition $\mathcal{P}$ of X defined by

$$r_\xi(C) := \sum_{x \in C} \xi(x) \quad \text{for any } C \in \mathcal{P}. \tag{3}$$

If the signal is a grayscale image, a rule can be a special type of blurring or downsampling of the image. If we roll that biased die with ξ in (1), the following three rules r, r', r'' (of ξ)

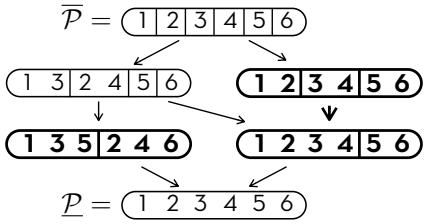
$$r(\{1,2\}) = 0.3, \qquad r(\{3,4\}) = 0.4, \qquad r(\{5,6\}) = 0.3 \tag{4}$$

$$r'(\{1,2,3,4\}) = 0.7, \qquad r'(\{5,6\}) = 0.3 \tag{5}$$

$$r''(\{1,3,5\}) = 0.6, \qquad r''(\{2,4,6\}) = 0.4 \tag{6}$$

correspond to the three partitions in (2). More generally, we define a *rule (regardless of any signal)* over a set X by any signal on any partition of X; we denote the set of all rules over X by $\mathcal{R}_X := \bigcup_{\mathcal{P}\in\{\text{all partitions of } X\}} \mathcal{S}_{\mathcal{P}}$.

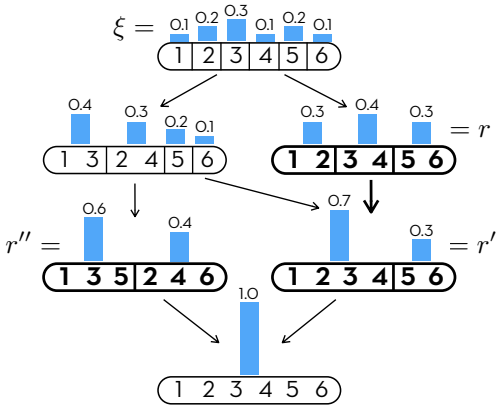
Partition lattice. Abstractions are hierarchical: one coarse-grained view can be coarser than another. Let the *partition lattice* $(\mathfrak{P}_X, \preceq)$ of a set X be the partially ordered set (poset) containing all partitions of X equipped with the partial order *coarser than* ($\preceq$), or *finer than* ($\succeq$), defined in the standard way. Let $\overline{\mathcal{P}} := \{\{x\} \mid x \in X\}$ and $\underline{\mathcal{P}} := \{X\}$ denote the finest and the coarsest partition, respectively. Per general lattice theory (Davey & Priestley, 2002), $\mathfrak{P}_X$ is a complete lattice: every subset $\mathfrak{P} \subseteq \mathfrak{P}_X$ has a unique supremum $\vee\mathfrak{P}$ and a unique infimum $\wedge\mathfrak{P}$, where $\vee\mathfrak{P}$ is called the *join* of $\mathfrak{P}$ denoting its coarsest common refinement and $\wedge\mathfrak{P}$ is called the *meet* of $\mathfrak{P}$ denoting its finest common coarsening. Considering our dice-rolling example, we have $\{\{1,2,3,4\},\{5,6\}\} \preceq \{\{1,2\},\{3,4\},\{5,6\}\}$, both of which are incomparable with $\{\{1,3,5\},\{2,4,6\}\}$. One can further check that the following DAG gives the smallest lattice containing the three partitions (boldfaced) in (2):



- An edge $\mathcal{P} \to \mathcal{P}'$ denotes $\mathcal{P}' \preceq \mathcal{P}$ (or the coarsening direction)
- The youngest common ancestor denotes the join
- The oldest common descendant denotes the meet

Note: $\{\{1,3\},\{2,4\},\{5\},\{6\}\}$ is a new abstraction discovered during lattice completion—perhaps new knowledge heretofore unconsidered?

Information lattice. The *information lattice* $(\mathcal{R}_\xi, \Leftarrow)$ of a signal $\xi : X \to \mathbb{R}$ is the poset of all rules of ξ equipped with the partial order *more general than*: for any two rules $r, r' \in \mathcal{R}_\xi$, we say r is more general than r' (or r' is more specific), denoted $r \Leftarrow r'$, if $\text{domain}(r) \preceq \text{domain}(r')$. Continuing with the partition lattice in our dice-rolling example, we have the corresponding information lattice as shown below.



- An edge $r \to r'$ denotes $r' \Leftarrow r$ (or the generalization direction)
- An information lattice is **isomorphic** to its underlying partition lattice

Note: $\mathcal{R}_\xi \subseteq \mathcal{R}_X$, i.e., the set of all rules of a signal on X is included in the set of all rules over X. Further, an information lattice of a signal is isomorphic to the underlying partition lattice via *projection* defined below.

Projection and lifting. We introduce a pair of operators to model human-like rule realization and abstraction as up and down inference directions in an information lattice. Roughly speaking,

a projection projects a signal down to a rule;
a lifting lifts a rule, or several rules, up to a signal.

More specifically, for any signal $\xi \in \mathcal{S}_X$, we define the *projection* operator $\downarrow^{\xi}\colon \mathfrak{P}_X \to \mathcal{R}_\xi$ by letting $\downarrow^{\xi}(\mathcal{P})$ be the rule of ξ on $\mathcal{P}$, i.e.,

$$\downarrow^{\xi}(\mathcal{P}) := r_\xi, \quad \text{as defined in (3).}$$

In our dice-rolling example, r, r', r'' in (4)-(6) are three projections of the signal ξ in (1) to the three abstractions in (2), respectively. One can check that $\downarrow^{\xi}\colon (\mathfrak{P}_X, \preceq) \to (\mathcal{R}_\xi, \Leftarrow)$ is an isomorphism. Conversely, we define the *general lifting* operator $\Uparrow^X\colon \mathcal{R}_X \to 2^{\mathcal{S}_X}$ by letting $\Uparrow^X(r)$ denote the set of all signals that *satisfy* the rule r, i.e.,

$$\Uparrow^X(r) := \{\xi \in \mathcal{S}_X \mid \downarrow^{\xi}(\text{domain}(r)) = r\} \subseteq \mathcal{S}_X.$$

To make lifting unique and per Principles of Indifference (Eva, 2019), we introduce a special lifting $\uparrow^X(r)$ to pick the most "uniform" signal in $\Uparrow^X(r)$. Formally, define $\|\cdot\|_q : \mathcal{S}_X \to \mathbb{R}$ by $\|\xi\|_q := (\sum_{x\in X} \xi(x)^q)^{1/q}$. For any $\xi, \xi' \in \mathcal{S}_X$ satisfying $\|\xi\|_1 = \|\xi'\|_1$, we say that ξ is more uniform than ξ' (or ξ' is more deterministic) if $\|\xi\|_2 \le \|\xi'\|_2$. Then, we define the *(special) lifting* operator $\uparrow^X\colon \mathcal{R}_X \to \mathcal{S}_X$ by

$$\uparrow^X(r) := \operatorname*{argmin}_{\xi \in \Uparrow^X(r)} \|\xi\|_2 \quad \text{(computed by simply averaging).}$$

We make two extensions to lifting a single rule to the signal domain: (a) lift multiple rules; (b) lift to a rule domain $\mathcal{P}$ instead of the signal domain X, denoted by $\Uparrow^{\mathcal{P}}(r)$ or $\uparrow^{\mathcal{P}}(r)$. Accordingly, we write $\Uparrow := \Uparrow^X$ and $\uparrow := \uparrow^X$ by default. Further, given a set of partitions $\mathfrak{P} \subseteq \mathfrak{P}_X$, we write

$$\mathcal{R} = \downarrow^{\xi}(\mathfrak{P}) := \{\downarrow^{\xi}(\mathcal{P}) \mid \mathcal{P} \in \mathfrak{P}\} \subseteq \mathcal{R}_\xi$$

to denote a rule set (obtained by projecting the signal to multiple abstractions), and write

$$\Uparrow(\mathcal{R}) := \bigcap_{r\in\mathcal{R}} \Uparrow(r) = \{\eta \in \mathcal{S}_X \mid \downarrow^{\eta}(\mathfrak{P}) = \mathcal{R}\}$$
$$\uparrow(\mathcal{R}) := \operatorname*{argmin}_{\eta \in \Uparrow(\mathcal{R})} \|\eta\|_2$$

to denote the set of signals that satisfy all rules in $\mathcal{R}$ (obtained by general lifting multiple rules) and the most uniform one (obtained by special lifting multiple rules), respectively. In our dice-rolling example, consider

$$\mathfrak{P} = \{\ \{\{1,2,3,4\},\{5,6\}\},\quad \{\{1,3,5\},\{2,4,6\}\}\ \}$$

to be the set comprising the last two partitions from (2). The corresponding rule set $\mathcal{R} = \downarrow^{\xi}(\mathfrak{P}) = \{r', r''\}$ with r', r'' as defined in (5) and (6), respectively. The general lifting $\Uparrow(\mathcal{R})$ of this rule set is the set of all signals that satisfy r' and r'', i.e., any pmf η that satisfies

$$\eta(1) + \eta(3) + \eta(5) = 0.6; \quad \eta(1) + \eta(2) + \eta(3) + \eta(4) = 0.7$$

The special lifting $\uparrow(\mathcal{R})$ is the most uniform pmf satisfying r' and r'', i.e., the solution to the following optimization problem:

$$\begin{aligned} \underset{\eta \geq 0,\ \sum_{i=1}^{6} \eta(i) = 1}{\text{minimize}} \quad & \|\eta\|_2 \\ \text{subject to} \quad & \eta(1) + \eta(3) + \eta(5) = 0.6 \\ & \eta(1) + \eta(2) + \eta(3) + \eta(4) = 0.7 \end{aligned}$$

Computing a special lifting is essentially solving a MaxEnt problem (Jaynes, 1957; Good, 1963): it replaces Shannon entropy of η by $-\|\eta\|_2$ and further exploits lattice structure. Notably, $1 - \|\eta\|_2^2$ is indeed a generalized entropy (Tsallis entropy), but makes optimization easier. More computational details are given in Appendix C.

5. Information Lattice Learning (ILL)

We first formalize ILL as a single optimization problem, then solve it practically in two phases. Let $\xi : X \to \mathbb{R}$ be a signal we want to explain. By *explaining*, we mean to search for a rule set $\mathcal{R} = \downarrow^{\xi}(\mathfrak{P}) \subseteq \mathcal{R}_{\xi}$ such that: (a) $\mathcal{R}$ recovers ξ well, or $\mathcal{R}$ is *essential*; (b) $\mathcal{R}$ is *simple*.

We say a rule set $\mathcal{R}$ recovers the signal ξ exactly if $\uparrow(\mathcal{R}) = \xi$. Exact recovery may not always be achieved. Information loss can occur for two reasons: (a) insufficient abstractions, i.e., the join $\vee\mathfrak{P}$ is strictly coarser than $\overline{\mathcal{P}}$; (b) the preference for uniformity in defining special lifting is inappropriate. When recovery is not exact, we introduce a loss function $\Delta(\uparrow(\mathcal{R}), \xi)$ to measure the mismatch, with a smaller Δ indicating a more essential $\mathcal{R}$. Common choices of Δ include distance (e.g., ℓ_p distance) or divergence (e.g., KL divergence) functions.

We say a rule set $\mathcal{R}$ is simpler if it contains fewer and simpler rules. Formally, we want $\mathcal{R}$ *minimal*, i.e., each rule $r \in \mathcal{R}$ is indispensable so as to achieve the same $\uparrow(\mathcal{R})$. Also, we want each rule $r \in \mathcal{R}$ *informationally simple*, measured by smaller Shannon entropy $\mathsf{Ent}(r)$, so r is more deterministic (Falk & Konold, 1997), easier to remember (Pape et al., 2015) and closer to our common-sense definition of a "rule". Notably, the partial order renders a *tradeoff* between the two criteria: $r \Leftarrow r'$ implies r is dispensable in any $\mathcal{R} \supseteq \{r, r'\}$ but on the other hand $\mathsf{Ent}(r) \leq \mathsf{Ent}(r')$, so including more-specific rules makes the rule set small yet each individual rule (informationally) hard.

The fundamental problem. The formal definition of an ILL problem is: given a signal $\xi : X \to \mathbb{R}$, we want a simple rule set $\mathcal{R}$ that recovers ξ well, i.e.,

$$\begin{aligned} \underset{\mathcal{R} \subseteq \mathcal{R}_{\xi}}{\text{minimize}} \quad & \Delta(\uparrow(\mathcal{R}), \xi) \\ \text{subject to} \quad & \mathcal{R} \text{ is minimal} \\ & \mathsf{Ent}(r) \leq \epsilon \quad \text{for any } r \in \mathcal{R} \end{aligned} \tag{7}$$

The search space involves the full information lattice $(\mathcal{R}_{\xi}, \Leftarrow)$, or isomorphically, the full partition lattice $(\mathfrak{P}_X, \preceq)$. Yet, the size of this lattice, i.e., the Bell number $B_{|X|}$, scales

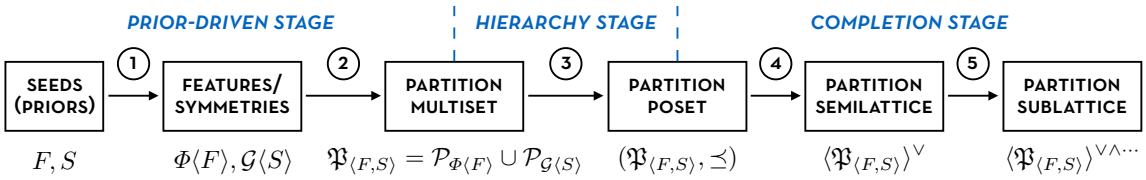
faster than exponentially in $|X|$. It is unrealistic to compute all partitions of X (unless X is tiny), let alone the partial order. Besides computational concerns, there are two reasons to avoid computing the full lattice (but to leave it implicitly in background): (a) the full lattice has unnecessarily high resolution, comprising many nearly-identical partitions particularly when X is large; (b) from the perspective of explainability, not every partition has an easy-to-interpret criterion by which the abstraction is made. As such, Formulation (7) is only conceptual and impractical. Next, we relax it and make it practical via two ILL phases.

5.1 Practical Lattice Construction: To Start Like a Baby (Phase I)

Information lattice construction plays a role similar to building a function class in ML, sometimes called meta-learning. While its importance is commonly understood, the construction phase in many data-driven models is often treated cursorily—using basic templates and/or *ad-hoc* priors—leaving most computation to the learning phase. In contrast, we put substantial computational and mathematical effort into our prior-driven construction phase. Pursuing generality and interpretability, we want *universal*, *simple* priors that are domain-agnostic and close to the innate cognition of a human baby (Marcus, 2018). Here we draw those from Core Knowledge (Spelke & Kinzler, 2007; Chollet, 2019), which include

- "the (small) natural numbers and elementary arithmetic prior" and
- "the elementary geometry and topology prior".

We then give algorithms to construct abstractions from these priors, and consider such a construction *prior-efficient* if it is interpretable, expressive, and systematic. In the flowchart below, we summarize the computation pipeline of information lattice construction as generating a partition sublattice.



① ② **Feature / Symmetry-induced partitions.** Unlike data clustering, our prior-driven partitions are induced from two data-independent sources—features and symmetries. We draw priors—in the form of seed features F and seed transformations S—from Core Knowledge as a basis, and then generate a set of partitions $\mathfrak{P}_{\langle F,S\rangle}$ as follows. Take $X = \mathbb{R}^2$ as an example,

$$F = \{w_{[1]}, w_{[2]}, w_{[1,2]}\} \cup \{\texttt{sort}, \texttt{argsort}, \texttt{sum}, \texttt{diff}, \texttt{div}_2, \ldots, \texttt{div}_{19}, \texttt{mod}_2, \ldots, \texttt{mod}_{19}\} \tag{8}$$

$$S = \{\text{horizontal, vertical, diagonal translations}\} \cup \{\text{rotations}\} \cup \{\text{reflections}\} \tag{9}$$

In (8), w_I denotes coordinate selection (like indexing/slicing in python) and the other functions are defined as in python (`div` and `mod` are from `divmod`). We further generate the

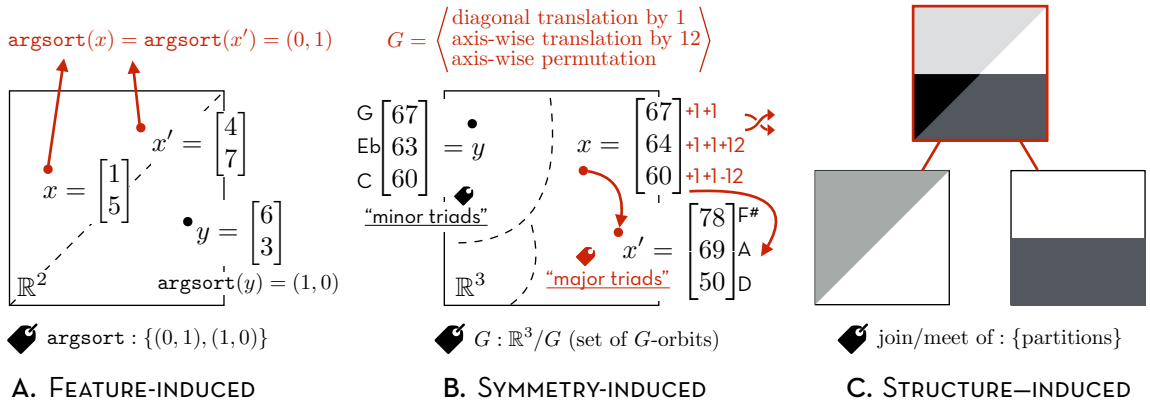


Figure 5: Three general ways of generating and tagging partitions: (A.) feature-induced via preimages; (B.) symmetry-induced via transformations; (C.) structure-induced via lattice completion. Features and symmetries are drawn or assembled from domain-agnostic priors.

following quantities along the construction pipeline.

	set of	generated by	via
$\Phi\langle F\rangle$:	features	F	function composition
$\mathcal{G}\langle S\rangle$:	subgroups	subsets of S	subgroup generation
$\mathcal{P}_{\Phi\langle F\rangle}$:	partitions	features in $\Phi\langle F\rangle$	preimages
$\mathcal{P}_{\mathcal{G}\langle S\rangle}$:	partitions	subgroups in $\mathcal{G}\langle S\rangle$	orbits

Then, $\mathfrak{P}_{\langle F,S\rangle} = \mathcal{P}_{\Phi\langle F\rangle} \cup \mathcal{P}_{\mathcal{G}\langle S\rangle}$, as a multiset, is the end result from Step ②. We sketch how to generate partitions from features and symmetries in Figure 5A and B, together with the third way of generating partitions via partial order, i.e., lattice completion, described below.

③ **Partition poset.** We sort $\mathfrak{P}_{\langle F,S\rangle}$ into poset $(\mathfrak{P}_{\langle S,F\rangle}, \preceq)$. We import our algorithmic skeleton from generic poset-sorting algorithms (Caspard et al., 2012; Daskalakis et al., 2011), with an outer routine incrementally adding elements and querying an inner subroutine (an oracle) for pairwise comparison. However, our poset is special, comprising *tagged partitions* where a tag records the generating source(s) of its tagged partition, e.g., features, symmetries. So, we specially designed both the outer routine Add_partition and the oracle Compare by leveraging (a) *transitivity* (valid for all posets), (b) partition *size* (valid for partitions), and (c) partition *tag* (valid for tagged partitions) to pre-determine and filter relations. We relegate implementation details to Appendix E. The data structures for posets include `po_matrix` and `hasse_diagram`, respectively encoding the partial order $\prec$ (ancestors/descendants) and the cover relation $\prec_c$ (parents/children) (Garg, 2015).

④ ⑤ **Partition semi/sublattice.** We complete $(\mathfrak{P}_{\langle F,S\rangle}, \preceq)$ into a lattice: the sublattice (of $\mathfrak{P}_X$) generated by $\mathfrak{P}_{\langle F,S\rangle}$. We use *alternating-join-and-meet* completions—one of the two generic sublattice-completion methods (Bertet & Morvan, 1999)—and discuss our choice and other related methods in Appendix D. Yet, we implement join-semilattice completion (meet-semilattice is dual) in our special context of tagged partitions, which reuses Add_partition in ③. The adjustments are (a) changing tags from features and symmetries to join formulae

and (b) changing the inner subroutine from pairwise comparison to computing join. We then run the completion sequence of alternating joins and meets, often stopping early to enhance interpretability. A single join or meet remains simple, interpretable as the intersection or union of concepts (e.g., the join of colored items and sized items gives items indexed by color and size), but multiple, alternating joins and meets may hinder comprehension. More implementation details on a single-step join-semilattice-completion, the completion sequence, and tips on early stopping are given in Appendix E.

5.2 Practical Lattice Learning: To Learn Like a Child (Phase II)

Learning in an information lattice means solving the optimization Problem (7), i.e., to search for a minimal subset of simple rules from the information lattice of a signal to best explain that signal. Let $\mathfrak{P}_\bullet$ be the sublattice (or semilattice, poset, if early stopped) from construction. Projecting a signal $\xi : X \to \mathbb{R}$ to $\mathfrak{P}_\bullet$ yields the information sublattice $\mathcal{R}_\bullet := \downarrow^\xi (\mathfrak{P}_\bullet) \subseteq \mathcal{R}_\xi$. It is worth reiterating that (a) $\mathfrak{P}_\bullet$ is constructed first and is data-independent; (b) ξ (data) comes after $\mathfrak{P}_\bullet$; (c) $(\mathcal{R}_\bullet, \Leftarrow)$ is isomorphic to $(\mathcal{P}_\bullet, \preceq)$: $\mathcal{R}_\bullet$ retains the partial order (`po_matrix` and `hasse_diagram`) and interpretability from $\mathcal{P}_\bullet$. As such, $\mathcal{R}_\bullet$ is what is given at the beginning of the learning phase.

The fundamental problem (relaxed). For practicality, we relax Problem (7): instead of the full lattice $\mathcal{R}_\xi$, we restrict the search space to $\mathcal{R}_\bullet$; instead of minimal rule sets, we consider only antichains (whose elements are mutually incomparable), necessary for minimality. This yields:

$$\begin{aligned} \underset{\mathcal{R}\subseteq\mathcal{R}_\bullet}{\text{minimize}} \quad & \Delta(\uparrow(\mathcal{R}), \xi) \\ \text{subject to} \quad & \mathcal{R} \text{ is an antichain} \\ & \mathsf{Ent}(r) \le \epsilon \quad \text{for any } r \in \mathcal{R} \end{aligned} \tag{10}$$

To solve Problem (10), we adopt a (greedy) idea similar to principal component analysis (PCA). We first search for the most essential rule in explaining the signal (i.e., the rule that decreases Δ most), then the second most essential rule in explaining the rest of the signal, and so on. Specifically, we start with an empty rule set $\mathcal{R}^{(0)} := \emptyset$, and add rules iteratively. Let $\mathcal{R}^{(k)}$ be the rule set formed by Iteration (Iter) k and $\mathcal{R}^{(k)}_\Leftarrow := \{r \in \mathcal{R}_\bullet \mid r \Leftarrow r' \text{ for some } r' \in \mathcal{R}^{(k)}\}$. Let $\mathcal{R}_{\le\epsilon} := \{r \in \mathcal{R}_\bullet \mid \mathsf{Ent}(r) \le \epsilon\}$. Then, in the $(k+1)$th iteration, we

$$\begin{aligned} \text{minimize} \quad & \Delta(\uparrow(\mathcal{R}^{(k)} \cup \{r\}), \xi) \\ \text{subject to} \quad & r \in \mathcal{R}^{(k)}_{feasible} := \mathcal{R}_{\le\epsilon} - \mathcal{R}^{(k)}_\Leftarrow. \end{aligned} \tag{11}$$

We pre-compute $\mathcal{R}_{\le\epsilon}$ (instead of the whole $\mathcal{R}_\bullet$) before iterations, which can be done by a breadth-first search (BFS) on $\mathfrak{P}_\bullet$'s `hasse_diagram`, from bottom (i.e., the coarsest) up. As to the monotonicity of Ent w.r.t. the partial order (cf. the grouping axiom of entropy (Cover & Thomas, 2012)), any BFS branch ends once the entropy exceeds ϵ. (For later use, we save the set $\underline{\mathcal{R}}_{>\epsilon}$ of ending rules in BFS, i.e., the lower *frontier* of $\mathcal{R}_{>\epsilon}$.) In contrast, $\mathcal{R}^{(k)}_\Leftarrow$ is computed per iteration (by querying $\mathfrak{P}_\bullet$'s `po_matrix`).

Nested vs. alternating optimization. Computing $\uparrow(\mathcal{R}^{(k)} \cup \{r\})$ is a minimization problem, making (11) a nested optimization: $\operatorname{argmin}_{r\in\mathcal{R}^{(k)}_{feasible}} \Delta(\operatorname{argmin}_{\eta\in\Uparrow(\mathcal{R}^{(k)}\cup\{r\})} \|\eta\|_2, \xi)$.

One may de-nest the two. Instead of comparing rules by lifting them up to the signal domain, we compare them "downstairs" on their own rule domains. So, instead of (11), we

$$\begin{aligned} \underset{r \in \mathcal{R}_{\leq\epsilon} - \mathcal{R}_{\Leftarrow}^{(k)}}{\text{maximize}} \quad & \Delta(\downarrow^{\uparrow(\mathcal{R}^{(k)})}(\text{domain}(r)), \downarrow^{\xi}(\text{domain}(r))) \\ & = \Delta(\downarrow^{\uparrow(\mathcal{R}^{(k)})}(\text{domain}(r)), r). \end{aligned} \tag{12}$$

The idea is to find the rule domain where the recovered signal $\uparrow(\mathcal{R}^{(k)})$ and the target signal ξ show the largest gap Δ. Adding this rule to the rule set zeros the largest gap in (12), and tends to minimize the original objective in (11). Nicely, in (12) the lifting $\uparrow$ does not involve r, so (11) is de-nested, which further iterates into an alternating min$\rightleftharpoons$max (or lift$\rightleftharpoons$project) optimization. Let $r_\star^{(k)}$ be the solution and $\Delta_\star^{(k)}$ be the optimal value in Iter k. We update $\mathcal{R}^{(k+1)} := \mathcal{R}^{(k)} \cup \{r_\star^{(k+1)}\} - \{r_\star^{(k+1)}\text{'s } \texttt{descendants}\}$ (so always an antichain), and go to the next iteration. Iterations end if the feasible set is empty, or may end early if the rule becomes less essential, measured by $|\Delta_\star^{(k+1)} - \Delta_\star^{(k)}| \leq \gamma$ in the nested setting, and $\Delta_\star^{(k)} \leq \gamma$ in the alternating setting (for some γ).

The full learning path & complexity. We denote a solve process for Problem (12) by $\textsc{Solve}(\epsilon, \gamma)$, or $\textsc{Solve}(\epsilon)$ if γ is fixed ahead. To avoid tuning ϵ manually, we solve for an ϵ-path. For $\epsilon_1 < \epsilon_2 < \cdots$, assume $\textsc{Solve}(\epsilon_i)$ takes K_i iterations, we run the following to solve the main relaxed Problem (10):

$$\emptyset = \mathcal{R}^{(0)} \to \textsc{Solve}(\epsilon_1) \to \mathcal{R}^{(K_1)} \to \textsc{Solve}(\epsilon_2) \to \mathcal{R}^{(K_1+K_2)} \to \cdots \tag{13}$$

In this way, lattice learning boils down to solving a sequence of combinatorial optimizations on the Hasse diagram of a lattice. We walk through the full process (13) via a toy example, starting with a signal $\xi : \{0, \ldots, 27\}^2 \to [0, 1]$ denoting an image of "7" and a toy-sized information lattice of the signal (Figure 6A). The sequence of optimizations (13) proceeds at two paces concurrently: the slower pace is indexed by ϵ_i; the faster is indexed by iteration number k. As mentioned earlier, the sets $\mathcal{R}_{\leq\epsilon_i}$ are pre-computed at the slower pace, with the $(i+1)$th BFS initialized from $\underline{\mathcal{R}}_{>\epsilon_i}$ (the ending rules in the ith BFS). The monotonicity of Ent w.r.t. the partial order assures that these BFSs add up to a single (global) BFS on the entire Hasse diagram, climbing up the lattice from the bottom. This is shown in Figure 6B as the monotonic expansion of the blue region ($\mathcal{R}_{\leq\epsilon}$) explored by BFS. Locally at each iteration along the slower pace, solving Problem (12) is quadratic in the worst case when the feasible set is an antichain (i.e., no order), and linear in the best case when the feasible set is a chain (i.e., totally ordered). Because local BFSs add up to a single BFS with standard linear complexity, the entire learning phase has a total complexity between linear and quadratic in the number of vertices and edges of the Hasse diagram. In general, the denser the diagram, the lower the complexity. This is because $\mathcal{R}_{\Leftarrow}^{(k)}$ tends to be large in this case with more descendants activated (i.e., red in Figure 6B), which in turn effectively shrinks the feasible set (i.e., the blue region minus red). For example, unlike the first three iterations in Figure 6B, the 4th iteration ($\epsilon = 3$) activates more than one rule, including the one being extracted as well as all its unexplored descendants. Further, the upper bound is rarely reached. Unlike in this toy example, BFS in practice is often stopped early when ϵ becomes large, i.e., when later rules become more random. Hence, when aiming for more

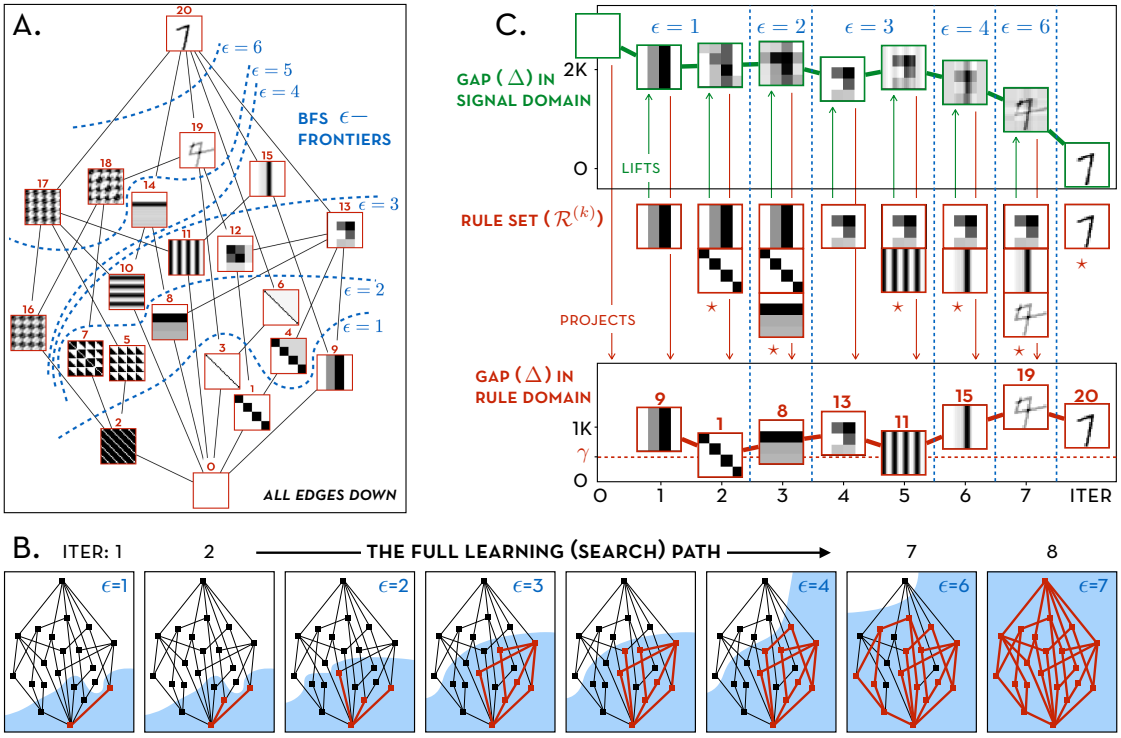


Figure 6: An ILL toy example: (A.) shows the Hasse diagram of an information lattice $\mathcal{R}_\bullet$, demarcated by six levels of ϵ-frontiers. (B.) outlines the full search process along the ϵ-solution path, where red highlights the activated rules and the blue background shows the marching of the BFS frontier. (C.) shows an ILL output—called a rule trace—where the up and down arrows show the alternating min⇌max (lift⇌project) optimization and each star marks the best rule set under an ϵ, i.e., a solution to the main relaxed Problem (10).

deterministic and disentangled rules only, *not all* vertices and edges are traversed by BFS. In the end of the learning process, for explanatory purposes, we store the entire ϵ-path and the $(\mathcal{R}^{(k)})_{k\geq 0}$ sequence instead of just the last one. This yields a *rule trace* as the standard ILL output, which we present below.

ILL output: rule trace. ILL outputs a rule trace comprising an evolving sequence of rules, rule sets, and recovered signals (Figure 6C). The three sequences are indexed by iteration and by ϵ-path, so the rule set by the last iteration under any ϵ (starred) is the returned solution to the fundamental Problem (10). For visualization convenience, we depict a rule of a 2D signal by its lifting (i.e., another grayscale image), because with pixels in the same cell colored the same way, we can use the lifting to sketch both partition and rule values. More precisely, when a lifting represents a rule, it is viewed by color blobs or superpixels; whereas a real lifting (i.e., a signal or a real image) is viewed normally by regular pixels. To clarify, all rules in Figure 6 are displayed in red boxes, whereas all liftings are in green. Figure 6C gives a full presentation of a rule trace. We also introduce a two-line shorthand, keeping only the sequence of the recovered signals and rules (Figure 7).

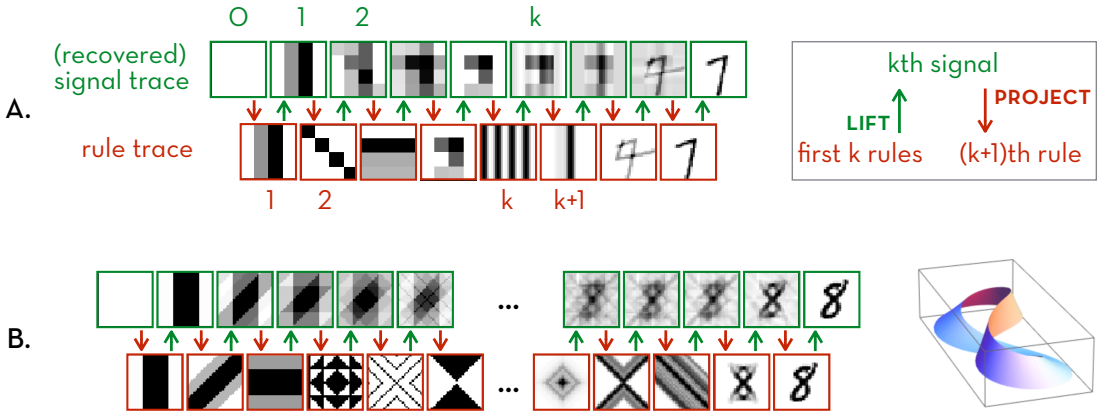


Figure 7: ILL's output in the format of two-line shorthand notation. (A.) shows a shorthand of Figure 6C: the kth signal along the signal trace (top line) is recovered by lifting up the first k rules along the rule trace (bottom line); then the $(k+1)$th rule is extracted by projecting the kth signal down to the lattice. (B.) shows an output in the same format for a different signal—the image of a figure "8"—revealing more symmetry patterns.

How to read a rule trace. A rule trace answers the fundamental question " what makes ξ an ξ ", or more formally,

what are the best ϵ-simple rules explaining ξ?

ILL rules are more interpretable than simply eyeballing patterns for 2 reasons:

(a) *The interpretability of the trace* is manifest in its controllability via ϵ, γ: smaller ϵ for simpler rules and larger γ for more essential rules.

(b) *The interpretability of each rule* is gained from its partition tag, i.e., the criteria by which the abstraction is made.

A single tag may contain multiple generating sources viewed as different interpretations of the same rule abstraction, e.g., `mod` and translation, `sort` and permutation. Like different proofs of a theorem, a partition tag with multiple sources reveals equivalent characterizations of a structure and thus, more insights of the signal. In this way, tags are not only computationally beneficial for constructing lattices, but key for interpretation.

5.3 Case Study: Handwritten Digits (Figure 6 and 7)

Consider the example in Figure 6 and its rule trace in Figure 7A. The signal ξ is the image of a handwritten "7". A rule of ξ, or the projection of ξ to a partition of the 28×28 image grid, can be viewed as gathering "ink" within each partition cell. Accordingly, a lifting can be viewed as redistributing the gathered "ink" in each cell. Hence, we call this view the *ink model*. For a toy illustration, we draw a small number of features and symmetries to generate a poset ($\mathfrak{P}_\bullet$) of 21 partitions. Its isomorphic counterpart in the information lattice ($\mathcal{R}_\bullet$) is shown by the Hasse diagram in Figure 6A. On top of the Hasse diagram, we further

demarcate the frontiers of the sublevel sets ($\mathcal{R}_{\leq\epsilon}$) by six blue dashed curves. Note that in this tiny diagram, we have sketched a full range of sublevel sets, yet for large diagrams, sublevel sets are constructed for small ϵ-values only in a single-pass BFS.

Figure 6C illustrates a complete ILL process in the alternating setting, with lift⇌project signified by the green up-arrows and red down-arrows, respectively. During the learning process, ILL tries to minimize the gap in the signal domain (upstairs) through iterative eliminations of the largest gap in the rule domain (downstairs). The rule set $\mathcal{R}^{(k)}$ formed per iteration is presented in the middle of Figure 6C, which jointly shows the complete rule trace continuously progressing along the ϵ-path. The rule set in the last iteration under any ϵ (marked by $\star$ in Figure 6C) is the returned solution to the main relaxed Problem (10). This rule set is used to answer what makes ξ an ξ. For example, let r_j denote the rule with ID j (here a rule ID is the same as the partition ID, the unique identifier for partitions generated in the construction phase). Then, among all rules whose entropies are no larger than $\epsilon = 2$, the third rule set in the trace $\mathcal{R}^{(3)} = \{r_9, r_1, r_8\}$ best explains what makes ξ an ξ. If more complex rules are allowed, say if all rule entropies are now capped by $\epsilon = 6$, $\mathcal{R}^{(7)} = \{r_{13}, r_{15}, r_{19}\}$ is the best.

We do not simply eyeball the rules to gain intuitive understandings, but use tags to achieve precise interpretation. For example, r_{19} in Figure 6 comes from a symmetry tag—a permutation subgroup—representing a reflection invariance. Rules r_8 and r_9 come from two feature tags—$\texttt{div}_7 \circ w_{[1]}$ and $\texttt{div}_7 \circ w_{[2]}$—respectively representing the continuous and even collapsing in the first and second coordinate (cf. the horizontal or vertical strips in either case). Both rules are later absorbed into r_{13} tagged by $\texttt{div}_7 \circ w_{[1,2]}$, since its rule domain is strictly finer. These rules (r_8, r_9, r_{13}) apparently summarize the horizontal and vertical parts of the handwritten "7". Further, the vertical part of the "7" is longer and slants more, so we see more vertically-patterned rules in the rule trace (r_9, r_{11}, r_{15}). These rules are obtained from finer and finer abstractions along the horizontal direction, so as to capture more details on the vertical part of that "7" such as its slope. Notably, among these vertically-patterned rules, r_{11} is induced from the symmetry representing a horizontal translation invariance, but it is quickly absorbed into r_{15} whose entropy is not much higher. This transient appearance of r_{11} implies that it plays a less important role in explaining this handwritten "7". From more experiments, symmetries play a less important role in explaining many "7"s. This is not the case in explaining many "8"s, however, where symmetries occur much more often. For example, consider a symmetry fused from translation and permutation invariances whose fundamental domain is homeomorphic to a Möbius strip. We hypothesize that this topological property might be related to the twisted nature of an "8". For a visual comparison, we present the rule traces learned from a "7" and an "8" in Figure 7, as well as the visual similarity between a Möbius strip and an "8".

6. Use Cases of ILL: Knowledge Discovery

For visual convenience, we used grayscale images in the above example to walk through our model, but if image recognition is our concern, learning interpretable rules for characterizing writing digits may not be of independent interest, as long as predictions are accurate. We note that it may be of separate interest in calligraphy or for understanding the nature of human writing systems (Changizi et al., 2006). Nevertheless, in a later section (Section 8)

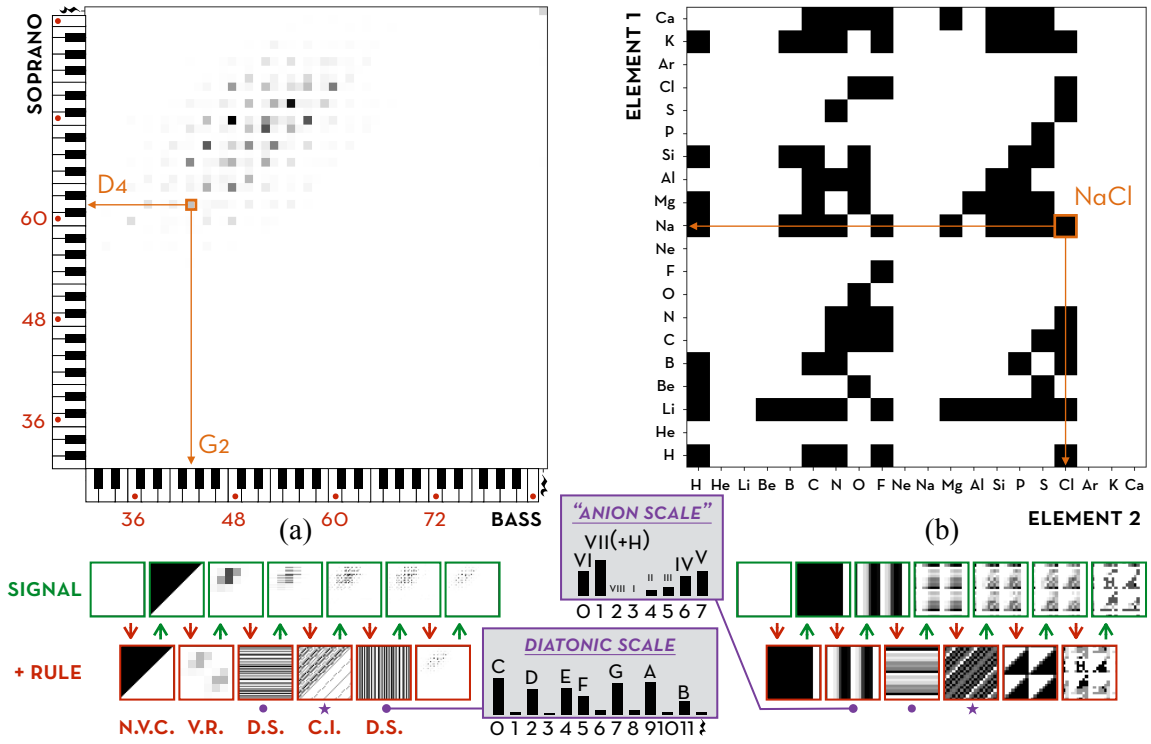


Figure 8: Music and chemical signals (top) with their rule traces learned from ILL (bottom).

we will demonstrate how ILL may be used, together with other ML models, to solve image classification problems. In this section, we suggest typical use cases of ILL for knowledge discovery in art and science.

We illustrate two ILL examples: learning music theory from scores and learning chemical laws from compounds. For these two illustrations, we fix the same priors—F, S in (8)(9)—thus the same lattice. We fix the same parameters: ϵ-path is $0.2 < 3.2 < 6.2 < \cdots$ (tip: a small initial offset, e.g., 0.2, is used to achieve nearly-deterministic rules) and γ is 20% of the initial signal gap. This fixed setting is used to show generality and for comparison. Yet, the parameters can be fine tuned in practice. As an overview, we first display the signal and rule trace from both subject domains in Figure 8.

Music illustration. Signals are probability distributions of chords encoded as vectors of MIDI keys. Figure 8a shows such a signal—the frequency distribution of two-note chords extracted from the soprano and bass parts of Bach's C-score chorales (Illiac Software, Inc., 2020)—with the learned rule trace listed below. In Figure 9, we present a zoomed-in view of some rules in the trace. The first rule is tagged by $\texttt{argsort} \circ w_{[1,2]}$ and has probability concentrated in one cell whose elements have a larger 1st-coordinate or S-coordinate (the black region above the diagonal). This is a deterministic rule that echoes the law of "no voice crossing (N.V.C.)", i.e., soprano is always higher than bass. Checking later rule tags reveals laws of voice range (V.R.), diatonic scale (D.S.), and consonant interval (C.I.)—almost all of the main static rules for two-voice counterpoint. Notably, the third rule is tagged by both $\texttt{mod}_{12} \circ w_{[1]}$ and vertical translation invariance. From both feature and symmetry views,

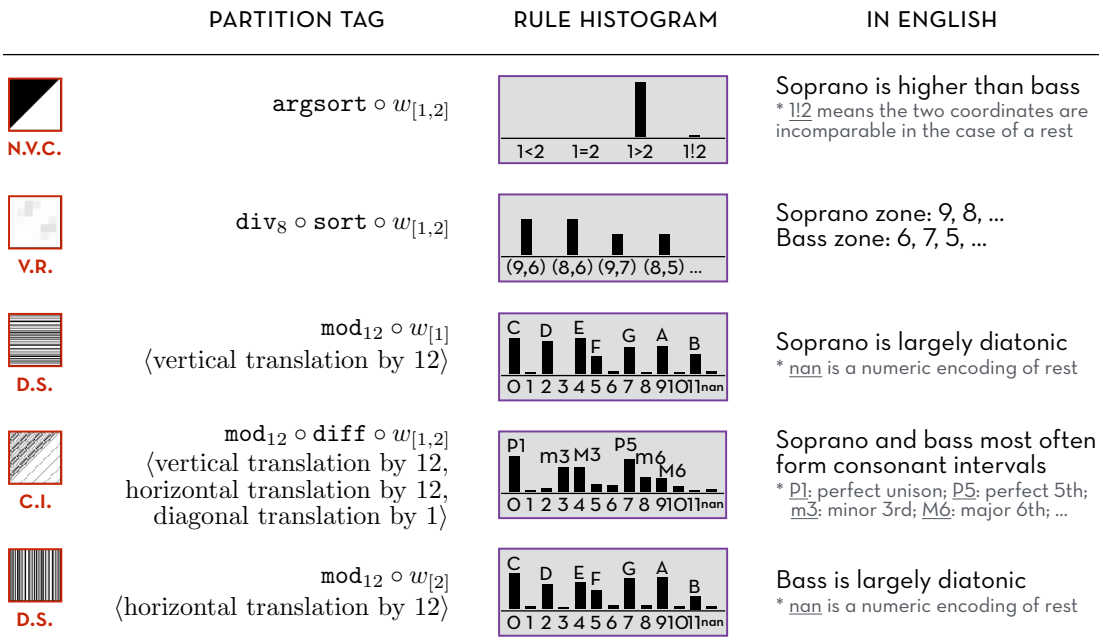


Figure 9: Zoomed-in views of ILL discovered music rules in Figure 8. The English translation of a rule's math form requires domain knowledge, but discovering the rule does not.

this tag identifies the concept of all Cs, all Ds, etc., which is the music concept of pitch class. The feature view explicitly reveals a period of 12 in pitches—the notion of an octave (in defining pitch class); the symmetry view reveals the topology—the manifold where the concepts lie—e.g., a 2D torus implied from the symmetries in the 3rd and 5th rule.

Chemistry illustration. Signals are Boolean-valued functions indicating the presence of compound formulae encoded as vectors of atomic numbers in a molecule database. Figure 8b shows a signal attained by collecting two-element compounds from the Materials Project database (Jain et al., 2013). In Figure 10, we give a zoomed-in view of some rules in the trace. The first rule tagged by $\mathtt{div}_{18} \circ w_{[2]}$ is deterministic: Element 2 cannot be Ar, K, Ca. It nicely captures the visual pattern in Figure 8b (check the last three vacant columns) and hints suggestively at some chemistry rules about both noble gas and nomenclature. The second rule tagged by $\mathtt{mod}_8 \circ w_{[2]}$ manifests peaks at cells tagged by feature values $1, 0, 7, 6$. These cells, for Element 2, are halogens (VII) including H, chalcogens (VI), pnictogens (V), crystallogens (IV). The third rule shows alkali metals (I), alkaline earth metals (II), crystallogens (IV), icosagens (III) are the cells common for Element 1. The next rule hints at common combinations, e.g., alkali metals (I) and halogens (VII), alkaline earth metals (II) and chalcogens (VI). These rules align well with known rules of forming compounds, e.g., for ionic/covalent bonding, and the ordering principles in chemical nomenclature.

Link Between Music and Chemistry. Note that the 2nd, 3rd, 4th rules for chemistry and the 5th, 3rd, 4th rules for music share nearly the same tags, with the main difference being that $\mathtt{mod}_{12}$ becomes $\mathtt{mod}_8$—period changes from 12 (a music octave) to 8 (number of main groups). So, when two chemical elements form a compound, they are like two music notes forming a chord! The musical concepts of pitch classes and intervals parallel the

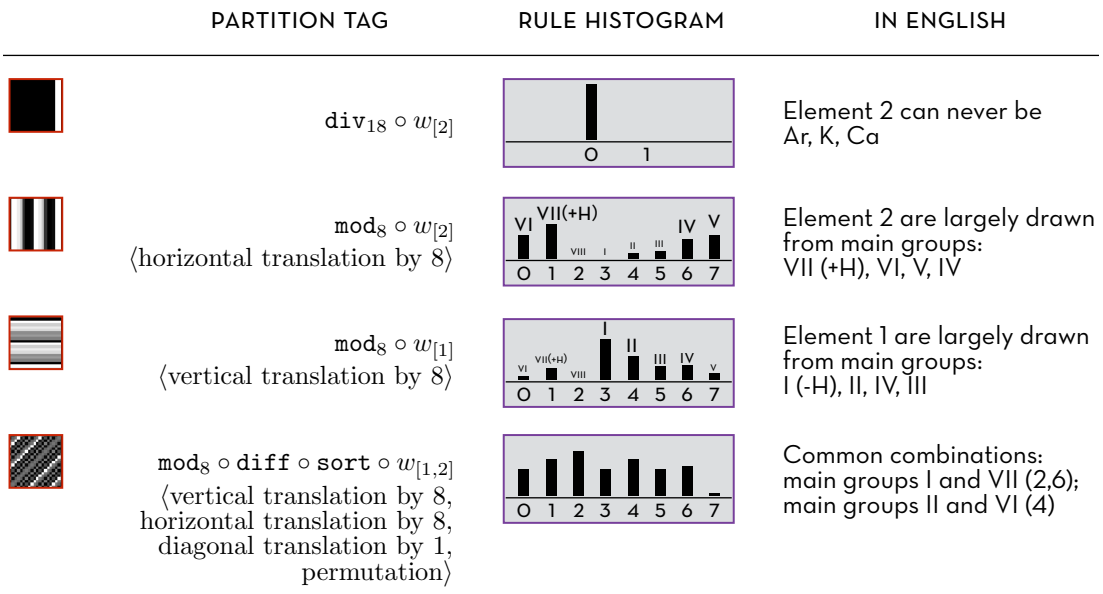


Figure 10: Zoomed-in views of ILL discovered chemistry rules in Figure 8.

chemical concepts of groups and distances. Although abstractions are shared, rules differ. Instead of a diatonic scale in Bach's chorales, chemistry uses a "cation scale" and an "anion scale". It is interesting that our intention to show ILL's generality by using the same lattice, parameters for different disciplines also suggests links between art and science by interpreting phenomena in one subject from the perspective of the other (Bodurow, 2018).

In the next section, we present a real-world application that builds atop our music illustration. Applications that extend the chemistry illustration are ongoing, with goals to not only recover the periodic table (Zhou et al., 2018), but also discover new chemistry laws, interpretations of existing laws, and materials.

7. Benchmark and Assessment on a Real-World Application

We extended our music illustration with a real application, which builds an automatic music theorist to teach student personalized lessons on music composition (Yu et al., 2016; Yu & Varshney, 2017). It specially implements the alternating min⇌max setting into a "student⇌teacher" model: the student is a (music) generator and the teacher is a discriminator. The two components form a loop where teacher guides student towards a target style through iterative feedback and exercise, which map onto our process of extracting and applying rules. This application reaches beyond our above music illustration by further considering:

- more *musical voices (or channels)*, so now signals are in higher dimensions and rules are on more complex chord structure;
- *temporal structure*, so now signals include various (un)conditional chord distributions (i.e., n-grams for $n = 1, 2, \ldots$), yielding both context-free and context-dependent rules, but new challenges too, namely *rare contexts/conditionals* and *contradictory rules*.

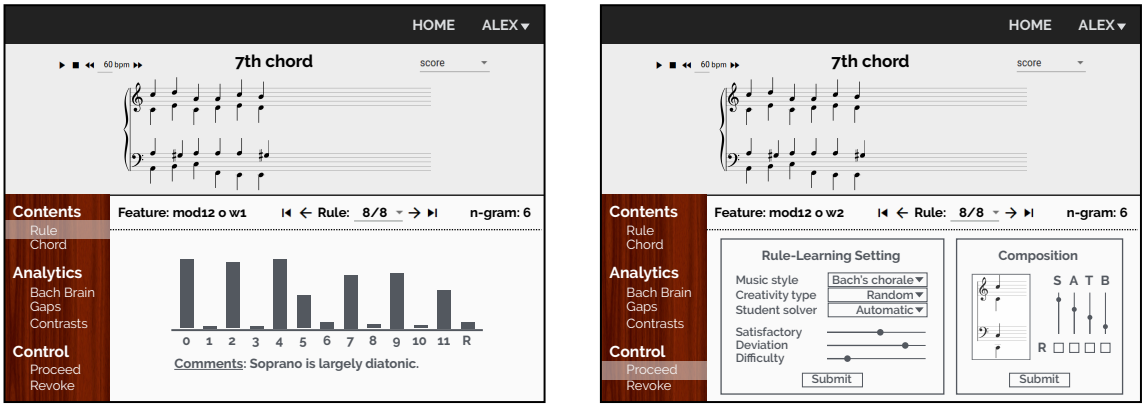


Figure 11: The music web interface: rule histogram (left); user control panel (right).

ILL's core idea of *abstraction* makes rare context common (more in "abstracted n-gram" below) and a redesigned lifting operator solves contradiction (Yu et al., 2017).

User interface. We designed a web interface (Figure 11) for the music application to facilitate user control over the rule-learning process and display of learned results. Users learn music rules, each rendered as a histogram over a tagged partition (i.e., machine-codified music concepts), and personalize their learning pace via provided knobs in the interface. These include rule difficulty, satisfactory level (a high value indicates high fidelity of the recovered signal to the original), and deviation level (a high value indicates a larger perturbation to the rule). Their set values are automatically converted to internal parameters (e.g., ϵ, γ, Laplace smoothing).

Abstracted n-grams (project n-grams to lattices). Music is highly contextual. To model context, we consider more than one signal simultaneously, including multiple n-grams with varying n's and varying conditionals. In this way, ILL projects n-grams to lattices and aims for rules that characterize not only individual chord formation but also melodic and harmonic progression. Accordingly, ILL produces both context-free and context-dependent rules, each indexed by 1) a partition and 2) a conditional under that partition. For example, given a partition that abstracts chords into roman numerals and conditioned on the previous two chords being $\mathrm{I}_4^6 \to \mathrm{V}$, an ILL rule specifies the probability distribution of the next roman numeral (*not* the next chord), and in this case reproduces the music rule on Cadential-64. Note that in a context-dependent rule, not only is the query chord abstracted, but also the conditional. This *abstracted n-gram* differs from plain n-gram models. The latter may suffer from rare context, i.e., a conditional occurs very few or even zero times in the training set. Yet, the core idea of abstraction makes small data large and rare contexts common. Under appropriately discovered equivalence classes, rare things become prevalent: "Although I have never seen this exact chord progression before, I have seen this type". Figure 12 exemplifies two context-free rules and a context-dependent one. These rule histograms are generated by ILL based on 370 of Bach's four-part chorales.

Evaluations. Via the music application's web interface, we conduct two studies to evaluate two important metrics for measuring performance of an ILL application, or a knowledge-discovery task in general, namely *rule-learning capability* and *human-interpretability*. The

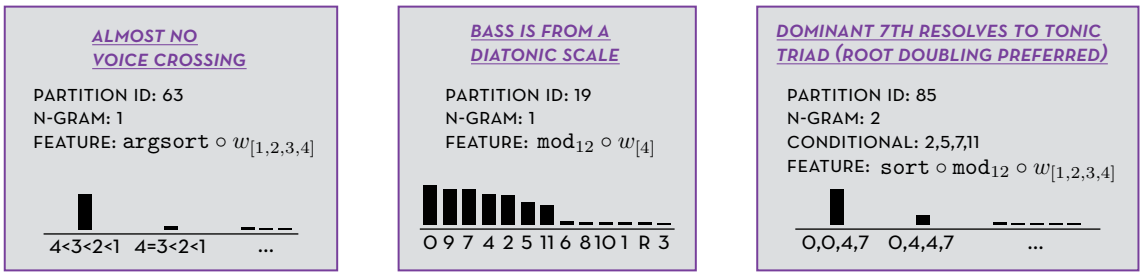


Figure 12: Examples of ILL rules: context-free (first two); context-dependent (last).

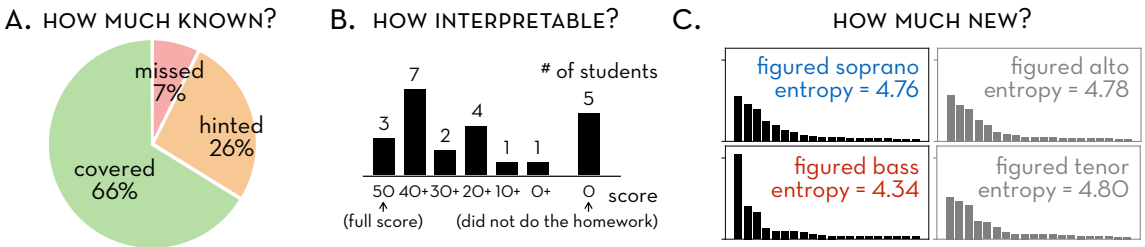


Figure 13: Assessments of ILL on knowledge-discovery tasks. (A.) Trained on 370 chorales, ILL explicitly reproduced 66% and implicitly hinted at 26% of a music theory curriculum. (B.) In an interpretation-focused assignment given to a CS+Music class, the majority of the students (excluding the five who did not do the assignment) were capable of interpreting most of the ILL-discovered rules. (C.) ILL revealed a new way of building chords, namely figured soprano, which is confirmed independently by other music theorists. More examples of new rules are presented in the main text.

former assesses both how much known knowledge an AI can reproduce which is common in automatic knowledge discovery settings (Schmidt & Lipson, 2009; Udrescu & Tegmark, 2020; Liu & Tegmark, 2020), and how much new it can discover. The latter normally involves human-subject study, which is common in the HCI and XAI communities. In our musical setting, we first summarize the main results in Figure 13 and detail the two studies below.

7.1 Assessing Rule-Learning Capability

To assess rule-learning capability, we compare machine-discovered rules with human-codified domain knowledge to identify how much known can be reproduced and how much new can be discovered. In our context, we compare ILL-distilled rules to a standard undergraduate music theory curriculum. The initial idea is to use known theory as a benchmark. Yet, the ultimate goal is not to use known theory as supervision to reconstruct only what we know, but also to discover new rules, new understandings of existing rules, new composition possibilities, and to teach rules in a personalized way.

Main setting. We are, *a priori*, aware of three major differences between human-codified music theory and ILL-generated rules.

(a) Music raw representation (input): known music theory is derived from all aspects of sheet music whereas ILL-generated rules are currently derived only from MIDI pitches and their durations. This is because we currently study ILL as a general framework. In later work, one can include more raw musical information, e.g., spelling, meter, measure, beaming, and articulation.

(b) Rule format (output): known music theory and ILL-generated rules have two different styles, with the former being more descriptive and absolute (hard), whereas the latter being more numerical and stochastic (soft). For instance, a music rule that strictly bans consecutive fifths is softened in an ILL rule that assigns a small non-zero probability to the undesired event. So, while it is possible to "translate", with information loss, a probabilistic rule in ILL to a verbal rule in known theory, it may not make sense to "translate" in the opposite direction. Further, it is unwise to hardcode known rules as categorical labels in a supervised setting, as music rules are inherently flexible and hardcoding may lead to a rule-based AI that generates "mechanical" music like the Illiac Suite (Hiller & Isaacson, 1957).

(c) Data set specific: known music theory is often intended for education purposes, rather than to reflect the style of a musical oeuvre. For instance, while consecutive fifths are banned in homework and exams, they are used in real-world song writing. Even in our data set of Bach's chorales, which are supposed to follow the known rules quite well, we see Bach himself wrote some consecutive perfect intervals. ILL-generated rules are specific to the input data set. We may find data sets that follow the known rules quite well (e.g., Bach's chorales), but others that break known rules and set their own.

Keeping these three differences in mind and isolating them from the comparison results, we reveal the remaining differences due to the rule-learning process itself. To come up with the benchmark, we compiled a comprehensive syllabus of laws from music theory taught in the Illinois School of Music's theory review course, which runs through the content of the full series of theory classes. This human-codified music knowledge is organized as a running list of 75 topics and subtopics indexed by lecture number. On the other hand, ILL-generated rules are indexed by partition (ID) and n-gram (n).

Results and analyses. Results are summarized in Table 1, where the colored crosses in the last column indicate topics missed by ILL for different reasons. Among the total 75 topics in Table 1, we first ignore 7 of them (red crosses) which require music raw representations beyond MIDI pitches and durations, such as accents and enharmonic respellings of some augmented sixth chords. ILL covered 45 out of the remaining 68 topics, yielding a coverage of 66%. Among the 23 missed topics, 18 (blue crosses) are related to deeper-level temporal abstractions such as harmonic functions, key areas, and forms. These temporal abstractions may be better modeled as *abstractions of transitions*, which are implicitly captured but not explicitly recovered from our current multi-abstraction multi-n-gram language model, modeling only *transitions of abstractions*. The other 5 missed topics (black crosses) are tricky and require *ad-hoc* encodings, not explicitly learnable, but may be implicitly captured from our current ILL implementation. Accordingly, the composition of the $30 = 7 + 18 + 5$ uncovered topics suggest three future directions to raise ILL's rule-learning capacity: (a) include more music raw representations; (b) model abstractions of transitions; (c) make music-specific adjustments when developing music apps or develop a more expressive and

general framework. Recall, however, that the goal here is not to reproduce what we know but to augment it. We may stop after enabling abstractions of transitions, which in the best case can yield an improved coverage of 84% (i.e., 93% of the topics from MIDI notes only) which may be sufficient for meaningful understanding and training.

Lecture	*Music Theory*	*Partition IDs*	*n-gram*	
1	music accents			✗
2	pitch	1-4	1	✓
2	pitch class	16-19	1	✓
2	interval	31-36	1	✓
2	interval class	97-102	1	✓
3	stepwise melodic motion (counterpoint)	1-4	2	✓
3	consonant harmonic intervals (counterpoint)	97-102	1	✓
3	beginning scale degree (counterpoint)	16-19	2	✓
3	ending scale degree (counterpoint)	16-19	2	✓
3	beginning interval class (counterpoint)	97-102	2	✓
3	ending interval class (counterpoint)	97-102	2	✓
3	parallel perfect intervals (counterpoint)	97-102	2	✓
3	directed perfect intervals (counterpoint)			✗
3	law of recovery (counterpoint)	1-4	$\geq$3	✓
3	contrapuntal cadence (counterpoint)	1-4, 97-102	2,3	✓
3	melodic minor ascending line (counterpoint)			✗
4	triads and seventh chords	26-30	1	✓
4	triads and seventh chords: quality	140-144	1	✓
4	triads and seventh chords: inversion	113-117	1	✓
5	figured bass	113-117	1,2	✓
5	roman numerals	81-85,129-133	1	✓
6	melodic reduction (Schenkerian analysis)			✗
7	passing tone (tones of figuration)	1-4, 134-144	3	✓
7	neighbor tone (tones of figuration)	1-4, 134-144	3	✓
7	changing tone (tones of figuration)	1-4, 134-144	4	✓
7	appoggiatura (tones of figuration)	1-4, 134-144	3	✓
7	escape tone (tones of figuration)	1-4, 134-144	3	✓
7	suspension (tones of figuration)	1-4, 134-144	3	✓
7	anticipation (tones of figuration)	1-4, 134-144	3	✓
7	pedal point (tones of figuration)	1-4	$\geq$ 3	✓
7	(un)accented (tones of figuration)			✗
7	chromaticism (tones of figuration)			✗
8	tonic (function)			✗
8	dominant (function)			✗
8	authentic cadence	1,4,81-85,129-133	2,3	✓
8	half cadence	81-85,129-133	2,3	✓
9	voice range (four-part texture)	1-4	1	✓
9	voice spacing (four-part texture)	31-41	1	✓
9	voice exchange (four-part texture)	20-25	2	✓
9	voice crossing (four-part texture)	53-63	1	✓
9	voice overlapping (four-part texture)			✗
9	tendency tone (four-part texture)	16-19	1,2	✓
9	doubling (four-part texture)	86-91	1	✓
10	harmonic reduction (second-level analysis)			✗
11	expansion chord			✗
12	predominant (function)			✗

Table 1 (cont.)

Lecture	*Music Theory*	*Partition IDs*	*n-gram*	
13	phrase model			✗
14	pedal or neighbor (six-four chord)	4,113-117	3	✓
14	passing (six-four chord)	4,113-117	3	✓
14	arpeggiated (six-four chord)			✗
14	cadential (six-four chord)	85,113-117,133	3,4	✓
15	embedded phrase model			✗
16	non-dominant seventh chord (function)			✗
17	tonic substitute (submediant chord)			✗
17	deceptive cadence (submediant chord)	81-85,129-133	2,3	✓
18	functional substitute (mediant chord)			✗
19	back-relating dominant	81-85,129-133	2,3	✓
20	period (I)			✗
21	period (II)			✗
22	period (III)			✗
23	applied chords (I)	81-85,129-133	2,3	✓
24	applied chords (II)	81-85,129-133	2,3	✓
25	applied chords (III)	81-85,129-133	2,3	✓
26	modulation (I)			✗
27	modulation (II)			✗
28	binary form (I)			✗
29	binary form (II)			✗
30	modal mixture			✗
31	Neapolitan	81-85,129-133	1	✓
32	Italian sixth chord	140-144	1	✓
32	French sixth chord	144	1	✓
32	German sixth chord			✗
32	Swiss sixth chord			✗
33	ternary form			✗
34	sonata form			✗

Table 1: Comparison of ILL-generated rules to human-codified laws of music theory taught in standard undergraduate music theory courses. Checks (45) in the last column denote topics recovered by ILL. Red crosses (7) denote topics not recoverable from our music raw representations; blue crosses (18) denote topics not recoverable from our n-gram transitions of abstractions/partitions; black crosses (5) denote topics not recoverable from the constructed lattice of abstractions. Partition IDs $63, 19, 85$ are exemplified in Figure 12.

From another source of music theory considering music symmetries (Tymoczko, 2010), we compare ILL-generated rules with a set of commonly used music operations, known as the OPTIC operations, namely octave shifts (O), permutations (P), transpositions (T), inversions (I), and cardinality changes (C). Results are summarized in Table 2, which shows that ILL covers the major four types of operations (OPTI). The music C operation is not recovered because it is not a transformation in the mathematical sense. Notations: t_v denotes a translation by the translation vector v, i.e., $t_v(x) := x + v$; r_A denotes a rotation (can be proper or improper) by the rotation matrix A, i.e., $r_A(x) := Ax$. As a special type of rotation matrix, $P^{(\cdots)}$ denotes a permutation matrix where the superscript is the cycle notation of a permutation. Note that ILL, as a general framework, considers a much larger universe of generic symmetries (from Core Knowledge) beyond those already considered in music. Therefore, ILL can not only study existing music symmetries, but also suggest new symmetries to be exploited in new styles as well as musical interpretations of symmetries discovered in other fields like chemistry.

Operation	*Music Description*	*Subgroup*	
Octave shift	"Move any note into a new octave."	$\langle\{t_{12\mathbf{e_1}}, t_{12\mathbf{e_2}}, t_{12\mathbf{e_3}}, t_{12\mathbf{e_4}}\}\rangle$	✓
Permutation	"Reorder the object, changing which voice is assigned to which note."	$\langle\{r_{P^{(1,2)}}, r_{P^{(2,3)}}, r_{P^{(3,4)}}\}\rangle$	✓
Transposition	"Transpose the object, moving all of its notes in the same direction by the same amount."	$\langle\{t_{\mathbf{1}}\}\rangle$	✓
Inversion	"Invert the object by turning it 'upside down'."	$\langle\{r_{-I}\}\rangle$	✓
Cardinality Change	"Add a new voice duplicating one of the notes in the object."		✗

Table 2: Comparison of ILL's symmetry-induced rule abstractions to music OPTIC.

New musical discoveries. We mention a few new rules discovered by ILL that piqued the interest of our colleagues from the Illinois School of Music.

(a) Tritone resolution plays a key role in tonal music and is an epitome in many more general harmonic resolutions. However, in Bach's chorales, tritones sometimes do not resolve in typical ways, but rather consistently transition to other dissonances such as a minor seventh, behaving like a harmonic version of an escape or changing tone.

(b) A new notion of "the interval of intervals" has been consistently extracted in several ILL-generated rule traces. This "second derivative", like acceleration in mechanics, might suggest a new microscopic chord structure heretofore unconsidered.

(c) New symmetry patterns reveal new harmonic foundations, hence new composition possibilities. As a parallel concept of harmony traditionally built on *figured bass* (indeed the dominant pattern in Bach's chorales confirmed by ILL), ILL reveals "figured soprano" as the next alternative in explaining Bach's music (Figure 13C). Although not the best view for explaining Bach according to ILL and is indeed not included in any standard music theory class, it may be a more efficient perspective to view or create music starting deviating from classical (e.g., in Jazz). This vision coincides with that from Casey Sokol (Sokol, 2016), a music professor at York University: "The idea of Figured Soprano is simply a way of taking this thinking from the top-down and bringing it into greater prominence as a creative gesture. So these exercises are not anything new in their ideation, but they can bring many new ideas, chord progressions and much else. It's a somewhat neglected area of harmonic study and it's a lot of fun to play with."

7.2 Assessing Human-Interpretability

To assess human-interpretability, we ask humans to interpret machine-generated rules. To identify to whom and to what degree ILL-generated rules are interpretable, we assess human-generated verbal interpretations of ILL rules rendered as sophisticated symbolic and numeric objects. We detail the collection and assessment procedure below.

Collecting human interpretations. The experiment was conducted in the form of a two-week assignment for 23 students. Students came from an undergraduate CS+Music degree program. Entry-level knowledge of computer science, math, and music theory is assumed from every student. In this setting, by *interpretability*, we mean interpretable to these students. We note, however, that all students are new to our AI system: none had read any ILL-generated rules before. The assignment contained three parts.

Part I provided detailed instructions on the format of rules as exemplified in Figure 12, including both feature-related and probability-related instructions. Symmetries were excluded from tags because group theory is an unfamiliar subject to these students. More specifically, we provided verbal definition, mathematical representation, and typical examples for each of the following terms: chord, window (for coordinate selection), seed feature, feature, rule, n-gram, histogram, data set (see Appendix F). A faithful understanding of these eight terms was the only prerequisite to complete the assignment. The estimated reading time of instructions was approximately one hour. Once self-reading was complete, the students were ready to go to the second and third parts of the assignment.

Part II contained eleven 1-gram rules—a histogram specified by window and seed feature(s); Part III contained fourteen 2-gram rules—a histogram now specified by window, seed feature(s), and additionally, a conditional. Students were asked to write what they saw in each of the histograms guided by the following two questions:

- Does the histogram agree/disagree with any of the music rules and concepts you know (write in music-theoretic terms when possible)?
- Does the histogram suggest something new (i.e., neither an agreement nor a disagreement, with no clear connection to anything you know)?

Answers to each of the 25 rules came in the form of text, containing word descriptions that "decode" the histogram—a symbolic and pictorial encoding. Students were explicitly instructed that writing out a description that was a literal repetition of the histogram (e.g., taking a modulo 12 of a chord results in a 91.2% chance of being $0, 0, 4, 7$) was unacceptable: they must reveal the music behind the math. In fact, we made it clear to students that we only wanted qualitative descriptions. Students were specifically told to only pay attention to the relative values of the probabilities (e.g., what are most likely, more likely, or virtually impossible). This assignment was due in two weeks, during which we asked students to complete it independently.

Assessing human interpretations. The assignment was designed such that every rule histogram encoded at least one music concept or rule consistent with standard music theory. In addition, every histogram contained either one additional known music rule or something strange that either conflicted with a known rule or represented something new. We assigned two points per rule.

We made an initial rubric containing the music keywords used to describe every rule histogram. Because students' answers were in the form of qualitative text, to ensure credibility and fairness of the initial rubric, we held a discussion session after the assignment due date with all students as well as teaching staff. During that session, we went over all 25 rules individually. For each, we first announced keywords in the initial rubric and explained to students that these keywords would later be used to grade their assignment. In the discussion session, every student was encouraged to object to any of our announced keywords

or to propose new keywords accompanied with a convincing explanation. New or modified keywords commonly agreed upon were added to the initial rubric. By the end of the discussion session, we compiled a more inclusive rubric containing broadly accepted keywords. This rubric-generating process was transparent to all students. We also made it clear that in order to get a high score, every student should try their best to convince others that a certain keyword should be added to the rubric.

In the final step, we manually graded every student's answer sheet against keywords in the rubric and computed their scores. A summary of the students' performance is presented in Figure 13B. Excluding students who did not do the assignment, a major source of score deduction was from misunderstanding the n-gram (e.g., the probability of the present conditioned on the past was mistakenly interpreted as the probability of the past conditioned on the present). This may be largely due to unfamiliarity with n-gram models for new CS+Music students. Nevertheless, the majority of students (excluding those 5 students who did not do the assignment) were capable of interpreting most of the rules, which provides evidence for the causability of ILL.

From explainability to causability. *Explainability*, as defined by the XAI community, highlights technically decision-relevant parts of machine representations and models that have contributed to model performance. While explainability has been widely studied in XAI, it fails to capture the communication of explanations to a human model. This yields the notion of *causability*—the measurable extent to which an explanation reaches a certain level of causal understanding for a human expert (Holzinger et al., 2020, 2021). Our above study on assessing human interpretation on machine-generated explanations is an example of the step from explainability to causability. In particular, we obtained a 0.82 *system causability scale*—a numeric causability measure in $[0, 1]$ as defined by Holzinger et al. (2020). We relegate the detailed calculation in Appendix G.

8. From Knowledge Discovery to Classification

To this point, we have seen ILL as a self-contained framework to learn human-interpretable rules and its use in knowledge discovery. This differs from many classical ML tasks, e.g., in computer vision (CV) or natural language processing (NLP). ILL itself is not a classifier and hence not directly comparable to ML models via prediction-based benchmarks. However, ILL can be integrated into existing classifiers to enhance interpretability and accuracy, especially in the "small data" regime. While a full presentation and comparison of ILL-enhanced ML models are beyond the scope of this paper, here we show a simple case where ILL can be combined with a Nearest-Neighbor classifier to predict handwritten digits in the MNIST dataset from singular or very small sets of examples. Our ILL-integration maintains human-interpretability from Nearest-Neighbor, while outperforming state-of-the-art MNIST classifiers for extremely small training sets (1–10 per class). In particular, we compare to TextCaps, a capsule-network variant that is among the world's top in MNIST and leader in the "small data" regime (Jayasundara et al., 2019).

ILL mimics human experiential learning; ILL-integrated Nearest-Neighbor naturally mimics humans' experience-based decision making where a new case is treated similarly as the closest in one's prior experience. Yet, humans are much more efficient in experience matching compared to the Nearest-Neighbor algorithm. Human brains are flexible in

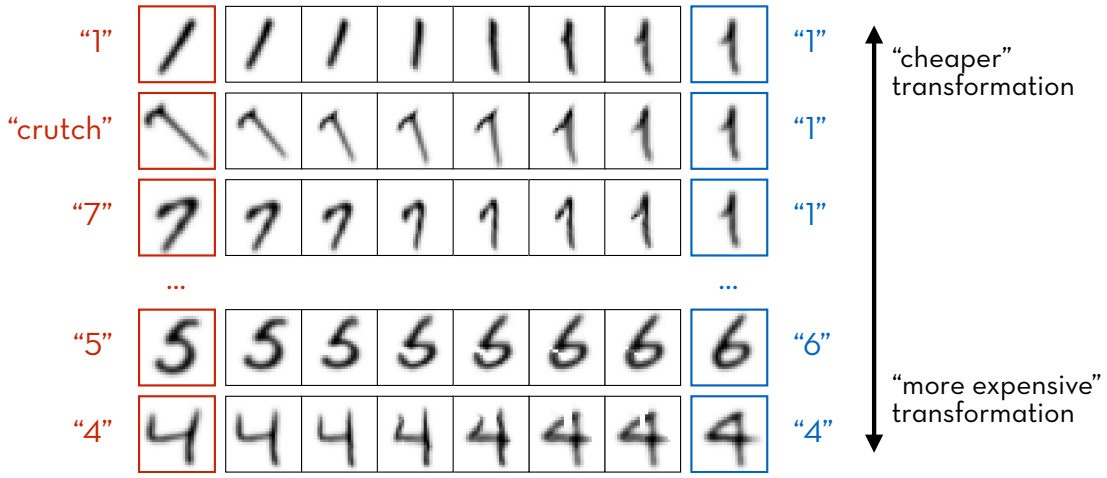


Figure 14: ILL discovers transformations that reflect what a human would naturally do to deform one figure into another. Notions of cheap and expensive transformations, as well as their induced small and large topological distances, also coincide with human intuition.

memorizing "proper" data representation and similarity in the *abstract* sense rather than rote memorization of raw data samples and distances in the *physical* sense. We do not memorize an image pixel by pixel or compare images pixel-wise. In this way, humans can generalize from only a few examples whereas the common Nearest-Neighbor often requires a dense sampling of the data space to perform well. For MNIST, it is hard to believe people must memorize thousands of training images per digit (e.g., $60,000$ in MNIST) in order to predict new digits well. Instead, persons may memorize a handful of examples and let the power of abstraction make "small data" large: when one sees a training example, it is not only a single, explicit example but an equivalence class suggesting many implicit examples.

For images, we consider symmetry-induced partitions. Between two images, we use ILL to learn a low-cost transformation, whose cost is further used to induce a distance function (for Nearest-Neighbor). Roughly, two images are similar if one can be transformed to the other with low cost. Below, we mention two distinguishing characteristics of this ILL variant.

- *Lattices of more generic symmetries.* Unlike common CV techniques (also our previous sections), here we do *not* restrict symmetry priors to pre-specified ones such as translation, rotation, or scaling invariances. Instead, we model the *smoothness* of any transformation—defined via *total variation*[1]—which is the only generic prior we use. Examples of smoother transformations include continuous deformations in an affine (e.g., isometries, scaling) style, but also piecewise affine or nonlinear ones (e.g., bending a "1" into a "7"). Smoothness prevents sudden changes, or shortcuts, e.g., wiping a "1" and regenerating a "7". It echoes the first category of Core Knowledge: "Objectness and elementary physics: objects move as continuous, connected, bounded wholes; objects do not suddenly cease to exist and do not suddenly materialize" (Chollet, 2019). Smooth

1. This is not the total variation of an image but rather the total variation of an image transformation.

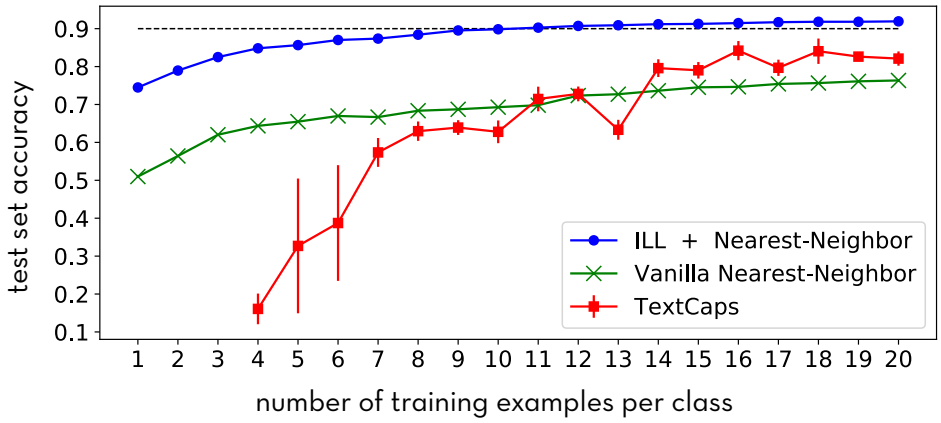


Figure 15: MNIST test-set accuracies of ILL-enhanced Nearest-Neighbor, vanilla Nearest-Neighbor (pixel-wise Euclidean distance), and TextCaps, using the first k ($k = 1, 2, \ldots, 20$) training examples per class. ILL-enhanced Nearest-Neighbor outperforms the other two models in all 20 cases, and is able to reach 90% test accuracy using only 10 training examples per class. TextCaps only runs when the training size ≥ 4 per class. Results from TextCaps are presented with error bars, each showing the mean and standard deviation over 5 independent runs (on the same training set).

transformations are consistent with human innate priors and natural for humans to understand. They are also efficient for abstracting out topological differences, as below.

- *Topological distance.* A distance function induced from symmetries abstracts the fundamental topological similarity between images. This has nothing to do with class labels and topologically similar images do not necessarily belong to the same class. In fact, neither of the images needs to be an image of a handwritten digit. An image of "1" may be closer to a "7", a letter "I", or even an image of a crutch. It is the later Nearest-Neighbor classifier that plays the role of supervision.

In Figure 14, we show some examples of "cheap" and "expensive" transformations, which abstract small and large topological distances, respectively. These coincide with human intuitions of deformation and similarity.

How does it perform in classification? We show in Figure 15 that the distance function learned from ILL can make the subsequent Nearest-Neighbor algorithm extremely data-efficient in MNIST classification (achieving nearly human level). The MNIST dataset is generally considered a "hello world" benchmark in CV, where many ML models have achieved nearly perfect prediction by fully leveraging the $60,000$ training images. Among those models, capsule networks have shown additional promise in working with small data sets. One particular variant, TextCaps, achieves a near state-of-the-art performance using 200 training examples per class (2000 in total). Yet, this is still not quite at human level: How about 10 per class? How about just 1, the case often faced by young children?

We test the performance of our ILL-enhanced Nearest-Neighbor and TextCaps, as well as the vanilla Nearest-Neighbor using pixel-wise Euclidean distance as baseline, in the regime of only a few training examples per class. With the training size growing from 1 image per class,

we run the three models on the same training set and collect their prediction accuracies on the entire MNIST test set.[2] In Figure 15, we see ILL-enhanced Nearest-Neighbor outperforms the other two models, reaching 90% test accuracy using only 10 training examples per class and nearly 75% accuracy with only 1. We note that TextCaps also does well on small data, catching up quickly when training size increases to a few dozen per class.

In Figure 15, training examples are selected as the first k ($k = 1, 2, \ldots, 20$) images in the training set per class, which may be viewed as a random sample. One may carefully select (by hand or by algorithm) a training subset comprising distinct "prototype" digits to mimic what humans might naturally do: observe more but select less to memorize. Using one such selected training subset consisting of only 51 images in total (i.e., $\sim$5 per class), ILL-enhanced Nearest-Neighbor can still achieve 90% test accuracy, while TextCaps' accuracy drops to $32.38\% \pm 2.48\%$.

We leave to later work full presentation of how to integrate ILL with existing ML models to improve interpretability and performance. Further, aligning ILL with high-performance yet black-box models can be of a particular interest, such as having an ILL version of deep learning or deep learning version of ILL. Besides performance and interpretability boost, ILL may be used to explain signals representing algorithmic behaviors to help people gain more insight about an algorithm. Consider studying AlphaGo or analyzing attention matrices learned from a Transformer-based NLP model like BERT or GPT (Rogers et al., 2020). Algorithmic insights gained can not only provide data scientists and engineers with more guidance in tuning a black-box model to its best performance, but also enable people to learn from high-performance AIs, such as acquiring new wisdom from AlphaGo (Shin et al., 2023).

9. Conclusion

Model transparency and interpretability are critical for trustworthy AI, especially when interacting directly with people, whether scientists, artists, or multidisciplinary researchers bridging *the Two Cultures* (Snow, 1959) like music and chemistry. The core philosophy underlying ILL arises from a human-centered standpoint and our pursuit of putting humanity back into artificial intelligence. We strive to develop human-like artificial intelligence (Zhu et al., 2020), which in turn may help advance human intelligence, a goal at the intersection of AGI (Goertzel & Pennachin, 2007), XAI (Adadi & Berrada, 2018), and AI as augmented intelligence (Jordan, 2019).

With this motivation, we have presented information lattice learning (ILL), a general framework that performs human-style rule learning. ILL addresses the fundamental question of "what makes X an X" and tackles human-interpretable knowledge discovery and human-like learning from small data. Our ILL framework is both prior-efficient and data-efficient, as manifest in its two development phases, respectively. With the construction phase capturing the core knowledge that babies are born with, followed by the learning phase mimicking human experiential learning, the entire ILL process epitomizes the idea of building AI that

2. The implementation borrowed from TextCaps' Github page requires at least 4 training examples per class to start with and it does not return the same result each time it runs on the same training and test sets. As such, results from TextCaps start from 4 training examples per class, each computed as the mean and standard deviation of 5 independent runs on the same training-test pair. TextCaps occasionally malfunctions, returning an accuracy of 10% or less—a random guess or worse. If this happens, we disregard the result and re-run TextCaps on the same training-test pair.

can "start like a baby and learn like a child." As such, ILL exhibits intrinsic interpretability, causability, and a higher-level generalization ability.

The key technology in ILL is grounded in theoretical foundations spanning group theory, lattice theory, information theory, and optimization. The mathematical formulation follows human intuition about abstraction and abstraction hierarchy, representing rules as coarsened signals and their lattice structure. ILL is widely applicable to knowledge discovery in diverse topic domains (e.g., music, chemistry), as well as data-scarce scenarios where many state-of-the-art machine learning algorithms fail. ILL can also be integrated with existing AI models to enhance both interpretability and performance, suggesting numerous future system designs.

10. Discussion: Limitations and Challenges

In this first effort, we develop ILL as a new, self-contained rule-learning framework to help people gain insight from sensory data. ILL is designed to be theoretically sound and intrinsically interpretable, behaving as a computational counterpart to human experiential learning. This paper illustrates setups and applications, but ILL is a general framework that admits new designs of its components, e.g., projection-and-lifting operators, priors, or lattice structures.

"Language" for interpretation. Designing a lattice not only sets the rule-learning capacity but also the "vocabulary" for interpretation which, like the Sapir-Whorf hypothesis for human language, shapes how a lattice explains signals. Likewise, priors have pros and cons based on what we seek to explain and to whom. Specifically, not all signals are best explained by symmetry, nor can everyone read symmetry equally well. One solution is to explore multiple lattices under diverse priors while balancing expressiveness and computation—a common practice in selecting between models.

Rules can be absent or insufficient. While one of the high-level ideas in ILL is to break the whole into simple pieces, whether a signal is actually governed by simple rules demands consideration. Sometimes, no rules exist and the signal is irreducibly complex. In such a case, ILL will recognize and indicate this (e.g., there is no ϵ-simple rule to describe that signal) and a *case-by-case* study will be needed. Sometimes, rules are insufficient: is music fully governed by music theory? Theory is better viewed as necessary but not sufficient for good music, as great composers need not be great theorists.

Benchmark/assessment in the long run. The two studies in Section 7 represent a move towards a systematic benchmarking and assessment paradigm for knowledge discovery. In the continuing effort to bridge human and machine intelligence, as well as to enhance human-AI interaction, new standardized procedures are needed for comparing machine-codified discoveries with human-codified knowledge and for using people to assess interpretability. Many challenges remain and will require long-term, consistent effort. We name a few below.

- Benchmarking is not as easy as for task-specific settings (Chollet, 2019), requiring better comparison schemes or a downstream task (e.g., classification).
- Assessments must focus on *new* discoveries, but not all new findings are equal. In general, how to make sense of new discoveries remains an open question. Interpreting new rules is fundamentally more challenging than asking experts to check-mark knowledge

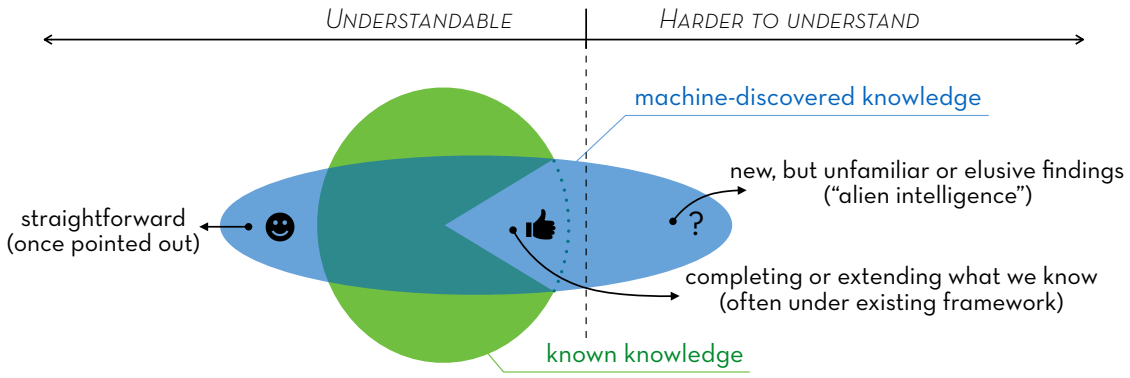


Figure 16: Different types of discovered rules: ILL can discover straightforward ones only unexpressed from their obviousness to nontrivial extensions of known knowledge. Furthermore, with an expansion of priors, it can generate novel, unfamiliar or even disorienting "alien" formulations.

from a textbook. Beside information metrics like entropy, the difficulty associated with interpreting a new rule is affected by familiarity with rule abstraction and how the new connects to the known. This opens a new dimension for exploring ILL abstraction priors to guide new discoveries towards different styles (Figure 16). Some argue that people may not want something new, but an expanded familiar (Eyal, 2015; Thompson, 2014). Others may be the opposite, such as appreciating an effective but "alien" move made by AlphaGo heretofore inconceivable by human masters (Shin et al., 2023). The former will yield knowledge discovery that completes existing understanding, as we do here. The latter discovers complementary or "alien" intelligence, which represents an exciting future direction.

- Evaluations should also focus on a human-centric view and test AI's ability to *assist* people in achieving their goals (augmented intelligence). Instead of a Turing test for machine-generated music, one might more productively conduct artistic evaluation between human-written music constructed with and without assistance from ILL.
- Further, illuminating rules associated with practice should not only help human students be better rule-followers, but more creative rule-breakers and rule-makers. Knowing machine-discovered rules can further unleash human creativity remains another long-term research question.

Acknowledgments

This work was funded in part by the IBM-Illinois Center for Cognitive Computing Systems Research (C3SR), a research collaboration as part of the IBM AI Horizons Network; the Andrew W. Mellon Foundation on "Algorithms, Models, and Formalisms"; and the National Science Foundation [#1829366].

Appendix A. Detailed Comparisons to Concept Lattice

Mathematically, let $(\mathfrak{P}_X, \preceq)$ be the partition lattice comprising all partitions of X and $(2^X, \subseteq)$ be the subset lattice comprising all subsets of X. Clearly, the power set 2^X is the same as $\{C \in \mathcal{P} \mid \mathcal{P} \in \mathfrak{P}_X\}$. That is, the subset lattice is also the lattice comprising all concepts from all partitions of X, which can be then called the full concept lattice. So, one can define any concept lattice in FCA as a sublattice of the full concept lattice (cf. Definition 3 in (Ganter et al., 2016)). Yet, such a concept sublattice does not have to include all concepts from a partition, and in many cases, it tends to miss many concepts if they are not known in the existing ontology. We give two examples below to further illustrate the connection between a partition lattice and a concept lattice.

First, consider biological taxonomy. Dogs and cats are two concepts in *species* which is an abstraction containing other concepts such as eagles. Likewise, mammals and birds are two concepts in *class* which is an abstraction containing other concepts such as reptiles and insects; further, animals and plants are two concepts in *kingdom*. In light of hierarchy, as abstractions, species $\succeq$ class $\succeq$ kingdom (in a partition lattice); as concepts, dogs $\subseteq$ mammals $\subseteq$ animals (in a concept lattice). Note that when forming a concept lattice, one may not need to include say, all species. Yet when having species as an abstraction in a partition lattice, this abstraction must contain all species including known species and unknowns, where the latter is usually of more interest for knowledge discovery.

Second, consider music theory. C major triads, C minor triads, and B diminished triads are concepts in an abstraction induced by music octave-shift and permutation invariance. Further, major triads, minor triads, and diminished triads are concepts in another abstraction induced by music octave-shift, permutation, and further transposition invariance. Clearly, for abstractions, the former abstraction is finer than the latter; for concepts, the set of C major triads is a subset (or a special case) of the set of major triads. However, chords that are not defined in traditional music theory but appear as new concepts in a known abstraction (e.g., the two above) may be more interesting, since they may suggest new composition possibilities while still obeying the same music abstraction, in this case the same music symmetry. New concepts from new abstractions may push the composition boundary even further, suggesting new types of chords discovered from e.g., new symmetry (but possibly within a known symmetry family). See the end of 7.1 for more examples from new discoveries.

Appendix B. Generalized Formalism for Information Lattice

The mathematical setting in the main paper is for a non-negative signal on a finite domain. However, this is not a limitation, but purely for notational brevity and computational reasons. First, regarding non-negativity, in many real scenarios, the signal is bounded and its value is only relative. In these cases, one can simply add an offset to the signal to make it non-negative. More generally, we can consider a signal to be any measurable function $\xi : X \to \mathbb{R}^n$. Then the notions of an abstraction, a concept, a rule, as well as the partial order can be generalized as in Table 3. Hence, the notion of an information lattice is still well-defined in the generalized setting. The essence of the two settings lies in how we formalize an abstraction, whether using a partition or a σ-algebra. However, the two are not very different from each other: any

partition of X generates a σ-algebra on X, and any σ-algebra on a countable X is uniquely generated by a partition of X (Çınlar, 2011).

	in the main paper	generalized
signal	$\xi : X \to \mathbb{R}$, X finite	$\xi : X \to \mathbb{R}^n$ measurable
abstraction	a partition $\mathcal{P}$ of X	a σ-algebra Σ on X
concept	a cell $C \in \mathcal{P}$	a subset $C \in \Sigma$
rule	$r : \mathcal{P} \to \mathbb{R}, r(C) := \sum_C \xi(C)$	$r : \Sigma \to \mathbb{R}^n, r(C) := \int_C \xi \, d\mu$
partial order	$\mathcal{P} \preceq \mathcal{P}'$	$\Sigma \subseteq \Sigma'$

Table 3: Formulations in the main paper and their generalizations.

Further, the main paper uses the sum in defining a rule of a signal, or the projection operator. However, other options are possible, e.g., mean, max, min, or a specially designed functional. The lifting operator can then be redesigned accordingly. In particular, besides always favoring the most uniform signal, the design of the special lifting can have extra freedom in considering other criteria for picking a signal from the general lifting.

Appendix C. More Insights on the Special Lifting

Consider the special lifting $\uparrow(\mathcal{R})$ for any rule set $\mathcal{R} = \downarrow^{\xi}(\mathfrak{P})$ of a given signal ξ. Computing $\uparrow(\mathcal{R})$ is simple if $\mathcal{R} = \{r\}$ contains only a single rule. In this case, $\uparrow(\mathcal{R})(x) = \uparrow(r)(x) := r(C)/|C|$ for any $x \in C \in \text{domain}(r)$, which requires simply averaging within each cell. However, computing $\uparrow(\mathcal{R})$ becomes much less trivial when $|\mathcal{R}| > 1$. By definition, we need to solve the minimization problem:

$$\uparrow(\mathcal{R}) := \text{argmin}_{r \in \Uparrow(\mathcal{R})} \|r\|_2. \tag{14}$$

Instead of directly throwing the above problem (14) into a generic optimization solver, there is a more efficient approach which also reveals more insights on the special lifting. More specifically, one can check that any multi-rule lifting $\uparrow(\mathcal{R})$ can be computed as a single-rule lifting $\uparrow(r^\star)$ where the single rule $r^\star$ is defined on the join $\vee\mathfrak{P}$ and is computed as follows:

$$r^\star := \text{argmin}_{r \in \Uparrow^{(\vee\mathfrak{P})}(\mathcal{R})} \|\tilde{r}\|_2, \tag{15}$$

$$\text{where the weighted norm } \|\tilde{r}\|_2 := \sqrt{\sum_C \frac{r(C)^2}{|C|}}.$$

So, instead of lifting $\mathcal{R}$ directly to the signal domain X, we lift $\mathcal{R}$ to the join $\vee\mathfrak{P}$ first and then to X. Since $|\vee\mathfrak{P}| \leq |X|$, the minimization problem (15) is in a smaller dimension compared to the original problem (14), and thus, can be solved more efficiently. In the minimization problem (15), by definition, $\Uparrow^{(\vee\mathfrak{P})}(\mathcal{R}) := \{r : \vee\mathfrak{P} \to \mathbb{R} \mid \downarrow^r(\mathfrak{P}) = \mathcal{R}\}$. Hence, every rule $r \in \Uparrow^{(\vee\mathfrak{P})}(\mathcal{R})$ can be treated as a single-rule summary of the rule set $\mathcal{R}$, and $r^\star$ is one of them—the one that yields the most uniform signal. Realizing the special lifting $\mathcal{R} \to \uparrow(\mathcal{R})$ as the two-step lifting $\mathcal{R} \to r^\star \to \uparrow(r^\star) = \uparrow(\mathcal{R})$ reveals the following insight: given rules abstracting ξ at different levels (coarser or finer), the best one can hope to *faithfully* explain ξ is at the level of the join. Determining ξ at any level finer than the join would then require additional assumptions other than the rule set itself, such as the preference of uniformity

used here. This further explains the two sources of information loss (join and uniformity) discussed in the recovery process of a signal (cf. Section 3 in the main paper). Notably, to determine a signal even at the level of join may be ambigious, since the general lifting $\Uparrow^{(\vee\mathfrak{P})}(\mathcal{R})$ to the join is not necessarily a singleton. This particularly implies that $r^\star$ as one of the single-rule summaries of $\mathcal{R}$ of ξ is not necessarily a rule of ξ, i.e., there is no guarantee that $r^\star = \downarrow^\xi(\vee\mathfrak{P})$. To make it so, we need more rules.

Appendix D. Existing Work on Sublattice Generation

General methods for computing the sublattice L_B of a full lattice L generated by a subset $B \subseteq L$ fall into two basic families, depending on whether the full lattice needs to be computed. The first uses alternating join- and meet-completions, with worse-case complexity $O(2^{|B|})$; the second characterizes the elements of L that belong to the sublattice, with complexity $O(\min(|J(L)|, |M(L)|)^2|L|)$ where $J(L)$ and $M(L)$ denote the number of join-irreducibles and meet-irreducibles, respectively (Bertet & Morvan, 1999). The latter requires computing the full lattice, which is intractable in our case of partition lattices, as $|L| = |\mathfrak{P}_X|$ grows faster than exponentially in $|X|$ whereas $|\mathfrak{P}_{\langle F,S\rangle}|$ is usually smaller than $|X|$. So, we use the first approach and compute alternating join- and meet-completions. The same principle of avoiding computing the full lattice has been applied to the special context of concept lattices (Kauer & Krupka, 2015), yet the technique there still requires the full formal context corresponding to the full concept lattice. Note that sublattice completion is, by definition, computing the smallest sublattice L_B (in a full lattice L) containing the input subset $B \subseteq L$, where L_B must inherit the meet and join operations from L. It generalizes but is not the same as Dedekind-MacNeille completion (Bertet & Morvan, 1999; MacNeille, 1937; Bertet et al., 1997).

Appendix E. Implementation Details on the Construction Phase

This section elaborates on the second half of Section 3.1 in the main paper, presenting more algorithmic details on poset construction and sublattice completion. The core data structures for posets are the so-called *adjacency matrix* and *Hasse diagram*, encoding the partial order $\prec$ and the cover relation $\prec_c$, respectively (Garg, 2015). The former is best for querying *ancestors* and *descendants* of a partition within the lattice; the latter is best for querying *parents* and *children* of a partition. (A more advanced technique includes chain-decomposition, but the two here are sufficient for this paper.) More specifically,

$$\begin{aligned} \mathcal{P}' \text{ is an ancestor of } \mathcal{P} &\iff \mathcal{P} \prec \mathcal{P}' \\ \mathcal{P}' \text{ is a parent of } \mathcal{P} &\iff \mathcal{P} \prec_c \mathcal{P}' \\ &\iff \mathcal{P} \prec \mathcal{P}' \text{ but no } \mathcal{P}'' \text{ satisfies } \mathcal{P} \prec \mathcal{P}'' \prec \mathcal{P}'. \end{aligned}$$

We introduce a few algorithmic notations. Given a partition poset $(\mathfrak{P}, \preceq)$, we use $\mathfrak{P}$.`po_matrix` and $\mathfrak{P}$.`hasse_diagram` to denote the adjacency matrix and Hasse diagram of $\mathfrak{P}$, respectively. For any partition $\mathcal{P} \in \mathfrak{P}$, we use $\mathcal{P}$.`ancestors`, $\mathcal{P}$.`descendants`, $\mathcal{P}$.`parents`, and $\mathcal{P}$.`children` to denote the sets of ancestors, descendants, parents, and children of $\mathcal{P}$, respectively. Notably, the two data structures are not only important for the construction phase but for the subsequent learning phase as well. The core subroutine in the construction

phase is Add_partition sketched as Algorithm 1. It is the key unit step in both poset construction and (join-)semilattice completion.

Poset construction. This corresponds to Step ③ in the flowchart in Section 3.1 of the main paper. Recall that poset construction refers to the process of sorting a multiset $\mathfrak{P}_{\langle F,S\rangle}$ of tagged partitions into a poset $(\mathfrak{P}_{\langle F,S\rangle}, \preceq)$, where the partition tags are features and symmetries. Naively, if we write an inner subroutine Compare$(\mathcal{P}, \mathcal{P}')$—called an *oracle* in the related literature—to compare two partitions, sorting a multiset into a poset amounts to $\binom{N}{2}$ calls of this pairwise comparison where N is the size of the input multiset. So, the common idea shared in almost all poset sorting algorithms is to reduce the number of oracle calls as much as possible. As mentioned in the main paper, considering the additional properties in our case, we leverage (a) *transitivity* (valid for all posets), (b) partition *size* (valid for partitions), and (c) partition *tag* (valid for tagged partitions) to pre-determine or pre-filter relations. In other words, we want to *infer from the context* as many pairwise relations as possible, so that the number of actual pairwise comparisons can be minimized.

More specifically, we start from an empty poset, and call Add_partition to incrementally add partitions from the input multiset to the poset. As the outer subroutine, Add_partition leverages transitivity and partition size by maintaining three live data structures, namely `size2partns`, `po_matrix`, and `hasse_diagram`, so as to avoid calling Compare whenever possible. Consequently, Compare is called only at two places (underlined in Algorithm 1): one for $=$ and one for $\prec$. When called as the inner subroutine, Compare$(\mathcal{P}, \mathcal{P}')$ does not always perform an actual computation for pairwise comparison. Instead, it first checks if the tags are informative (e.g., compositions/supergroups imply coarser partitions) and only if not, makes an actual comparison. With the additional information from partition size, an actual comparison can be done in $O(|X|)$ time via a mapping process. More specifically, given two partitions $\mathcal{P}, \mathcal{P}'$, without loss of generality, we assume $|\mathcal{P}| \leq |\mathcal{P}'|$. An actual comparison is made by tentatively creating a mapping $\nu : \mathcal{P}' \to \mathcal{P}$. One can check that such a ν exists if and only if $\mathcal{P} \preceq \mathcal{P}'$. Hence, if $|\mathcal{P}| = |\mathcal{P}'|$ (resp. $|\mathcal{P}| < |\mathcal{P}'|$), one can determine $=$ (resp. $\prec$) if ν is created successfully or incomparability otherwise. The mapping complexity is linear in $|X|$, with linear coefficient 1 if mapping succeeds and with linear coefficient < 1 if mapping fails. In the worst case (e.g., if all partitions are incomparable), all $\binom{N}{2}$ pairwise comparisons are required. Our algorithm works best when partitions are richly related (i.e., the Hasse diagram is dense), which is indeed the case for our tagged partitions induced from systematically formed features and symmetries.

Semilattice completion. This corresponds to Step ④ in the flowchart in Section 3.1 of the main paper. Recall that join-semilattice completion refers to the process of completing a partition poset into a semilattice. We only detail join-semilattice completion, since meet-semilattice completion can be done symmetrically. Formally, we want to compute the join-semilattice of $\mathfrak{P}_X$ generated by the input poset $(\mathfrak{P}_{\langle F,S\rangle}, \preceq)$. We denote the resulting join-semilattice by $\langle \mathfrak{P}_{\langle F,S\rangle} \rangle^{\vee}$. By definition, $\langle \mathfrak{P}_{\langle F,S\rangle} \rangle^{\vee} := \{\vee\mathfrak{P} \mid \mathfrak{P} \subseteq \mathfrak{P}_{\langle F,S\rangle}\}$. Naively, if computing $\langle \mathfrak{P}_{\langle F,S\rangle} \rangle^{\vee}$ literally from the above definition, one has to iterate over all subsets of $\mathfrak{P}_{\langle F,S\rangle}$ and compute their joins. This amounts to 2^N join computations where $N = |\mathfrak{P}_{\langle F,S\rangle}|$ is the size of the input poset, and moreover, many of the joins are not pairwise. Yet, similar to our earlier poset construction, we may reduce the computations of joins by an incremental method, which also embeds Add_partition as a subroutine and utilizes partition sizes and tags, but now the tags are join formulae instead of features or symmetries.

Algorithm 1: ADD_PARTITION $(\mathcal{P}_\tau, \mathfrak{P})$: adds a tagged partition $\mathcal{P}_\tau$ to a partition poset $(\mathfrak{P}, \preceq)$

Input: a tagged partition $\mathcal{P}_\tau$, where the tag τ can be a feature/symmetry or a join/meet formula; a partition poset $(\mathfrak{P}, \preceq)$, with the following members and hash tables:

- every $\mathcal{P} \in \mathfrak{P}$ is a unique partition (indexed by a unique identifier)
- $\mathfrak{P}$.`partn2tags[`$\mathcal{P}$`]` $:= \{\tau \mid \mathcal{P}_\tau = \mathcal{P}\}$ denotes the set of all tags inducing $\mathcal{P}$
- $\mathfrak{P}$.`size2partns[`k`]` $:= \{\mathcal{P} \mid |\mathcal{P}| = k\}$ denotes the set of all $\mathcal{P} \in \mathfrak{P}$ with size k
- $\mathfrak{P}$.`po_matrix` encodes the partial order $\prec$, best for getting $\mathcal{P}$.`ancestors`/`descendants`
- $\mathfrak{P}$.`hasse_diagram` encodes the cover relation $\prec_c$, best for getting $\mathcal{P}$.`parents`/`children`

Step 1: determine if $\mathcal{P}_\tau$ is new by COMPARE$(\mathcal{P}, \mathcal{P}_\tau)$ (for $=$) for every $\mathcal{P} \in \mathfrak{P}$.`size2partns[`$|\mathcal{P}_\tau|$`]`

if $\mathcal{P}_\tau \in \mathfrak{P}$.`size2partns[`$|\mathcal{P}_\tau|$`]`: update $\mathfrak{P}$.`partn2tags[`$\mathcal{P}_\tau$`]` by adding τ; **return**

else: create a new hash entry $\mathfrak{P}$.`partn2tags[`$\mathcal{P}_\tau$`]` $= \{\tau\}$; **proceed to Step 2**

Step 2: add the new partition $\mathcal{P}_\tau$ to $\mathfrak{P}$

(2a) update $\mathfrak{P}$.`size2partns[`$|\mathcal{P}_\tau|$`]` by adding $\mathcal{P}_\tau$

(2b) update $\mathfrak{P}$.`po_matrix` and $\mathfrak{P}$.`hasse_diagram`

- **for** every existing size $k < |\mathcal{P}_\tau|$ sorted in a descending order:
  - **for** every $\mathcal{P} \in \mathfrak{P}$.`size2partns[`$k$`]`:
    - **if** $\mathcal{P}$.`parents` $\cap$ $\mathcal{P}_\tau$.`descendants` $\neq \emptyset$: update $\mathfrak{P}$.`po_matrix` by adding $\mathcal{P} \prec \mathcal{P}_\tau$
    - **else**: COMPARE$(\mathcal{P}, \mathcal{P}_\tau)$; update $\mathfrak{P}$.`po_matrix` and $\mathfrak{P}$.`hasse_diagram` if $\mathcal{P} \prec \mathcal{P}_\tau$ (here one can check: it is necessarily the case that $\mathcal{P} \prec_c \mathcal{P}_\tau$)
- do the above symmetrically **for** every existing size $k > |\mathcal{P}_\tau|$ sorted in an ascending order
- (note: every $\mathcal{P} \in \mathfrak{P}$.`size2partns[`$k$`]` for $k = |\mathcal{P}_\tau|$ is incomparable with $\mathcal{P}_\tau$)
- clean cover relation: remove any $\mathcal{P}_* \prec_c \mathcal{P}^*$ from $\mathfrak{P}$.`hasse_diagram` if $\mathcal{P}_* \prec_c \mathcal{P}_\tau \prec_c \mathcal{P}^*$

More specifically, we start with an empty semilattice $\mathfrak{P}$, and add partitions in $\mathfrak{P}_{\langle F,S\rangle}$ to $\mathfrak{P}$ one by one from smaller-sized to larger-sized (note: the size information is maintained in $\mathfrak{P}_{\langle F,S\rangle}$.`size2partns`). When a partition $\mathcal{P} \in \mathfrak{P}_{\langle F,S\rangle}$ is to be added, we make a tag named by itself, i.e., let $\mathcal{P}_\tau := \mathcal{P}$ with $\tau := \{\mathcal{P}\}$, and then call ADD_PARTITION$(\mathcal{P}_\tau, \mathfrak{P})$. There are two possibilities here: $\mathcal{P}_\tau$ already exists in $\mathfrak{P}$ (call ends by Step 1) or $\mathcal{P}_\tau$ is new (call ends by Step 2). In the former, we are done with $\mathcal{P}_\tau$. In the latter, for every $\mathcal{P}' \in \mathfrak{P} \backslash \{\mathcal{P}_\tau\}$, compute the pairwise join $\mathcal{J}(\mathcal{P}') := \vee\{\mathcal{P}_\tau, \mathcal{P}'\}$ and its tags $\mathcal{T}(\mathcal{P}') := \{\tau \cup \tau' \mid \tau' \in \mathfrak{P}.\texttt{partn2tags[}\mathcal{P}'\texttt{]}\}$, and call ADD_PARTITION$(\mathcal{J}(\mathcal{P}')_{\mathcal{T}(\mathcal{P}')}, \mathfrak{P})$. Like COMPARE, computing join can be optimized by leveraging previously computed tags and partial order in the input poset $\mathfrak{P}_{\langle F,S\rangle}$, so as to avoid an actual join computation whenever possible. When inferring from the context is not possible, one can perform an actual join computation $\vee(\mathcal{P}, \mathcal{P}')$ in $O(|X|)$ time. This is done by collecting the unique pairs of cell IDs $(C(x), C'(x))$ for every $x \in X$, where $C(x)$ and $C'(x)$ denote the cell IDs of x in $\mathcal{P}$ and $\mathcal{P}'$, respectively. In the worst case (e.g., if all partitions

are incomparable and join-irreducible), the complexity is inevitably $O(2^N)$. However, like in poset construction, our algorithm works best when the partial order structure is rich.

Practical tips for sublattice completion. This corresponds to Step ⑤ in the flowchart in Section 3.1 of the main paper. Recall that constructing the sublattice of $\mathfrak{P}_X$ generated by $\mathfrak{P}_{\langle S,F\rangle}$ follows the alternating process: $L_0 := \mathfrak{P}_{\langle S,F\rangle}$, $L_1 := \langle L_0\rangle^\vee$, $L_2 := \langle L_1\rangle^\wedge$, $L_3 := \langle L_2\rangle^\vee$, and so forth, which terminates as soon as $L_{k-1} = L_k$. We denote the end result by $\langle \mathfrak{P}_{\langle S,F\rangle}\rangle^{\vee\wedge\cdots}$, which is the desired sublattice. However, we may want to stop early in the completion sequence, due to concerns from computation, interpretability, expressiveness, as well as their tradeoffs. We suggest a practical tip on deciding where to stop. If the input poset $\mathfrak{P}_{\langle F,S\rangle}$ is small, run alternating joins and meets, or even complete it to the sublattice if affordable. If $\mathfrak{P}_{\langle F,S\rangle}$ is moderate, complete the joins only (as join is closely related to rule lifting, see Appendix C for more details). If $\mathfrak{P}_{\langle F,S\rangle}$ is large, just use it.

Appendix F. Instructions on the Rule-Interpretation Assignment

We provided the following written instructions to the students, as the only prerequisite to complete the assignment. The instructions covered the basic terms that appear in the assignment, including both definitions and examples.

- Tetrachord

  Definition. A *tetrachord* is a chord consisting of four music pitches, including rests. We represent any tetrachord by a four-dimensional vector of MIDI numbers and use the special character `nan` (called not-a-number) to denote a rest. Every dimension of a chord vector denotes a voice: 1–Soprano, 2–Alto, 3–Tenor, 4–Bass.

  Examples. $c_1 = (60, 52, 77, 60)$, $c_2 = (32, \texttt{nan}, 69, \texttt{nan})$.

- Window

  Definition. A *window* is a list of indices (1-based) that select certain dimension(s).

  Examples. $w_1 = (1, 3, 4)$, $w_2 = (1,)$.

  - Apply the window w_1 (as a function) to the chord c_1 yields: $w_1(c_1) = (60, 77, 60)$.
  - $w_1(c_2) = (32, 69, \texttt{nan})$.
  - $w_2(c_1) = (60,)$.
  - $w_2(c_2) = (32,)$.

- Descriptor (Seed Feature)

  Definition. A *descriptor* (or a *seed feature*) is a function that performs an arithmetic operation on a chord.

  Examples. $\texttt{mod}_{12}$, `diff`, `sort`, `order` (i.e., a well-defined `argsort`).

  - $\texttt{mod}_{12}(c_1) = (0, 4, 5, 0)$.
  - $\texttt{mod}_{12}(c_2) = (8, \texttt{nan}, 9, \texttt{nan})$.
  - $\texttt{diff}(c_1) = (12, -25, 17)$.
  - $\texttt{diff}(c_2) = (\texttt{nan}, \texttt{nan}, \texttt{nan})$.

- $\texttt{sort}(c_1) = (52, 60, 60, 77)$.
- $\texttt{sort}(c_2) = (32, 69, \texttt{nan}, \texttt{nan})$.
- $\texttt{order}(c_1) =$ "$2 < 4 = 1 < 3$".
- $\texttt{order}(c_2) =$ "$1 < 3!2!4$" ("!" denotes incomparable).

Note. `nan` arithmetically behaves as python `numpy.nan`. More specifically, `nan%12==nan, {anything}-nan==nan, nan-{anything}==nan`, and when sorting, `nan` is always sorted towards the end.

- Feature

  Definition. A *feature* is a function which is the composition of a window function, and one or more descriptors.

  Examples. $w_1 * \texttt{mod}_{12} * \texttt{diff} * \texttt{sort}$. Apply this feature to c_1 yields:

$$\begin{aligned}
&\texttt{sort}(\texttt{diff}(\texttt{mod}_{12}(w_1(c_1)))) \\
=&\texttt{sort}(\texttt{diff}(\texttt{mod}_{12}((60, 77, 60)))) \\
=&\texttt{sort}(\texttt{diff}((0, 5, 0))) \\
=&\texttt{sort}((-5, 5)) \\
=&(-5, 5).
\end{aligned}$$

- n-gram

  Definition. An *n-gram* is a sequence of n consecutive chords. Given an n-gram, we query the probability distribution of the nth chord (i.e., the current chord) *conditioned* on the previous $(n-1)$ chords. Therefore,

  - in a 3-gram (notated as $\texttt{ngram} = 3$), we consider the probability distribution conditioned on the previous 2 chords;
  - in a 2-gram (notated as $\texttt{ngram} = 2$), we consider the probability distribution conditioned on the previous 1 chord;
  - in a 1-gram (notated as $\texttt{ngram} = 1$), we consider the *unconditional* probability distribution regardless of any previous chord(s).

  Examples. $\texttt{ngram} = 1$ (no conditional), $\texttt{ngram} = 2$ (contains a conditional, or `cond` in short, involving 1 previous chord), $\texttt{ngram} = 3$ (contains a `cond` involving 2 previous chords), and so forth.

  There are two special values for any conditional (`cond`) regardless of the feature:

  - `s` denotes the start of a piece;
  - `e` denotes the end of a piece.

  Therefore,

  - conditioned on `s` (or `cond = s`) means the current chord is the beginning chord;
  - the probability of `e` means the probability of the current chord being an ending.

- Rule

  Definition. Every *rule* in this assignment is represented by a histogram under

  - a given feature,
  - a given ngram, and
  - a given conditional (`cond`) if $\texttt{ngram} > 1$.

  Every feature is indexed by a unique id called `pid`; every feature (content) is represented by a window followed one or more descriptors. We provide the mapping between each `pid` and its content.

  Examples.

  $[\texttt{pid} = 68, \texttt{ngram} = 2, \texttt{cond} = 7]$ uniquely indexes a 2-gram rule.

  More specially, this is a 2-gram rule, with feature $\texttt{pid} = 68$. The content of this feature ($\texttt{pid} = 68$) is given as $(1, 4) * \texttt{diff} * \texttt{mod}_{12}$, i.e., first applying window $(1, 4)$ to a chord, and then descriptor `diff` followed by descriptor $\texttt{mod}_{12}$. So, one can read this rule mechanically (literally) as follows:

  - first, considering the feature $(1, 4) * \texttt{diff} * \texttt{mod}_{12}$ ($\texttt{pid} = 68$),
  - second, conditioned on the previous chord having a feature value 7,
  - the probability distribution of the top few most probable feature values of the current chord is given by the histogram.

  $[\texttt{pid} = 68, \texttt{ngram} = 2, \texttt{cond} = \texttt{s}]$ uniquely indexes another 2-gram rule.

  One can read this rule mechanically (literally) as follows:

  - first, considering the feature $(1, 4) * \texttt{diff} * \texttt{mod}_{12}$ ($\texttt{pid} = 68$),
  - second, conditioned on the previous chord having a feature value `s` (this means the current chord is the beginning chord of a piece),
  - the probability distribution of the top few most probable feature values of the current chord (in this case, the first chord) is given by the histogram.

  $[\texttt{pid} = 68, \texttt{ngram} = 1]$ uniquely indexes a 1-gram rule.

  One can read this rule mechanically (literally) as follows:

  - first, considering the feature $(1, 4) * \texttt{diff} * \texttt{mod}_{12}$ ($\texttt{pid} = 68$),
  - second, regardless of any previous chord(s),
  - the (unconditional) probability distribution of the top few most probable feature values of the current chord is given by the histogram.

  Note: we made it clear to the students that literally rephrasing a rule as above would not be counted as an interpretation—they must reveal the music behind the math.

- Data set

  370 Bach's 4-part C-score chorales.

- Form of the histograms

  Every histogram displays the top K most probable feature values (labels) and their corresponding probabilities. More specifically,

  - $K = \min\{50, \text{the total number of feature values with non-zero probabilities}\}$;
  - probabilities are rounded to two decimal places.

  So, every observed 0.00 is not exactly 0, but a positive value that is close to 0. Further,

  - when $K < 50$, every unseen feature value has a probability 0 (exact);
  - when $K = 50$, every unseen feature value has either a probability 0 (exact) or a probability that is even smaller than the 50th smallest probability shown in the histogram.

  Yet in any event, one should not bother too much with the actual value of a probability; comparing the probabilities relatively is good enough.

Given the above instructions on reading machine-generated rules (symbolic and numeric), this assignment asked students to write in words their interpretations (word descriptions in music-theoretic terms when possible) of every ILL rule, by answering two guiding questions: does the histogram (a) agree/disagree with any known music theory, or (b) suggest something new? Word descriptions that were literal repetitions of the histogram (e.g., taking a modulo 12 of a chord results in a 91.2% chance of being $0, 0, 4, 7$) was not acceptable; instead, qualitative descriptions revealing the insights are preferred (e.g., most likely a root-position C major triad).

Appendix G. System Causability Scale

Defined by Holzinger et al. (2020), the *system causability scale*, or *SCS*, is a numeric score measuring causability. SCS first uses the standard five-level Likert scale to score ten proposed items and then aggregates the results into a single score that is between 0 and 1. The ten items proposed by Holzinger et al. (2020) are quoted below.

1. I found that the data included all relevant known causal factors with sufficient precision and granularity.
2. I understood the explanations within the context of my work.
3. I could change the level of detail on demand.
4. I did not need support to understand the explanations.
5. I found the explanations helped me to understand causality.
6. I was able to use the explanations with my knowledge base.
7. I did not find inconsistencies between explanations.
8. I think that most people would learn to understand the explanations very quickly.
9. I did not need more references in the explanations: e.g., medical guidelines, regulations.
10. I received the explanations in a timely and efficient manner.

Rather than conducting a new psychometric experiment, we use our existing user study and further characterizations. We calculate SCS for our ILL system in the following table. Note that our user study in Figure 13B holistically assesses Items 2, 5, 6, 8, and hence, the same score is repeatedly used for these ratings.

Question	*Rating*	*How it is rated*
1. Factors in data	3	Figure 13A (66% known) and Figure 13C (a few new)
2. Understood	4	Figure 13B: $\frac{5\times10+4\times2+3\times4+2\times1+1\times1}{10+2+4+1+1} \approx 4.05$
3. Change detail level	5	Figure 11: users have control over all customized changes
4. Need teacher/support	5	Students did the task independently
5. Understanding causality	4	Figure 13B: $\frac{5\times10+4\times2+3\times4+2\times1+1\times1}{10+2+4+1+1} \approx 4.05$
6. Use with knowledge	4	Figure 13B: $\frac{5\times10+4\times2+3\times4+2\times1+1\times1}{10+2+4+1+1} \approx 4.05$
7. No inconsistencies	5	No inconsistencies were discovered by the students
8. Learn to understand	4	Figure 13B: $\frac{5\times10+4\times2+3\times4+2\times1+1\times1}{10+2+4+1+1} \approx 4.05$
9. Needs references	2	Need instructions/guidelines for ILL terms (Appendix F)
10. Efficient	5	ILL rules are automatically elicited within seconds
SCS	0.82	$\sum_i Rating_i/50$

Appendix H. Some Basics on Music-Theoretic Terms

We include a few basic music-theoretic terms and refer interested readers to any standard music theory textbook for more.

Pitch describes how high or low a musical note is played. Pitch represents the fundamental frequency of sound wave vibrations and is measured in Hertz or MIDI numbers.

Scale comes from the Latin word "ladder". A musical scale is a collection of notes within an octave arranged by their pitch. Major and minor scales are two common types of scales. There are seven scale degrees. The function of a scale degree relates to the amount of tension it creates. All the different pitches in major and minor scales include: tonic, supertonic, mediant, subdominant, dominant, submediant, leading tone (in major) or subtonic (in minor).

Interval depicts the distance between two notes. Intervals are measured in half steps, whole steps, and their position in the scale. A half-step interval is one semitone (or MIDI distance 1); two half steps make a whole step (or MIDI distance 2). Intervals are described in numbers of half steps between two notes. Intervals are also described by quality. The five interval qualities are major, minor, perfect, augmented and diminished.

Chords are the building blocks of harmony. A chord is a combination of two or more notes played together. Some of the common types of chords include triad and seventh chords. Common chord qualities include major, minor, augmented, and diminished. Chord inversions are variations of a chord. A chord inversion is created by transposing notes in a chord.

Chord progressions are ordered series of chords. There are many commonly used chord progressions, each creates a certain mood and/or emotion. Most popular songs use one of

only a handful of chord progressions. A (harmonic) cadence is a chord progression concluding a phrase, section, or piece of music.

References

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). *IEEE Access*, *6*, 52138–52160.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, *35*(8), 1798–1828.

Bertet, K., & Morvan, M. (1999). Computing the sublattice of a lattice generated by a set of elements. In *Proc. 3rd Int. Conf. Orders, Algorithms Appl.*

Bertet, K., Morvan, M., & Nourine, L. (1997). Lazy completion of a partial order to the smallest lattice. In *Proc. 2nd Int. Symp. Knowl. Retr., Use and Storage for Effic. (KRUSE 1997)*, pp. 72–81.

Bodurow, C. (2018). Music and chemistry—what's the connection?. *Chem. Eng. News*, *96*(36), 39–39.

Carvalho, D. V., Pereira, E. M., & Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, *8*(8), 832.

Caspard, N., Leclerc, B., & Monjardet, B. (2012). *Finite Ordered Sets: Concepts, Results and Uses*. No. 144 in Encyclopedia of Mathematics and its Applications. Cambridge University Press.

Chaitin, G. J. (1987). *Algorithmic Information Theory*. Cambridge University Press.

Changizi, M. A., Zhang, Q., Ye, H., & Shimojo, S. (2006). The structures of letters and symbols throughout human history are selected to match those found in objects in natural scenes. *Am. Nat.*, *167*(5), E117–E139.

Chater, N., & Vitányi, P. (2003). Simplicity: A unifying principle in cognitive science?. *Trends Cogn. Sci.*, *7*(1), 19–22.

Chollet, F. (2019). On the measure of intelligence. arXiv:1911.01547v2 [cs.AI].

Çınlar, E. (2011). *Probability and Stochastics*, Vol. 261. Springer Science & Business Media.

Clark, B. S., Stein-O'Brien, G. L., Shiau, F., Cannon, G. H., Davis-Marcisak, E., Sherman, T., Santiago, C. P., Hoang, T. V., Rajaii, F., James-Esposito, R. E., et al. (2019). Single-cell RNA-seq analysis of retinal development identifies NFI factors as regulating mitotic exit and late-born cell specification. *Neuron*, *102*(6), 1111–1126.

Cover, T. M., & Thomas, J. A. (2012). *Elements of Information Theory*. John Wiley & Sons.

Daskalakis, C., Karp, R. M., Mossel, E., Riesenfeld, S. J., & Verbin, E. (2011). Sorting and selection in posets. *SIAM J. Comput.*, *40*(3), 597–622.

Davey, B. A., & Priestley, H. A. (2002). *Introduction to Lattices and Order*. Cambridge University Press.

Eisen, M. B., Spellman, P. T., Brown, P. O., & Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. U.S.A.*, *95*(25), 14863–14868.

Eldar, Y. C., & Kutyniok, G. (2012). *Compressed Sensing: Theory and Applications*. Cambridge University Press.

Eva, B. (2019). Principles of indifference. *J. Philos.*, *116*(7), 390–411.

Evans, R., Hernández-Orallo, J., Welbl, J., Kohli, P., & Sergot, M. (2021). Making sense of sensory input. *Artif. Intell.*, *293*, 103438.

Eyal, N. (2015). People don't want something truly new, they want the familiar done differently. `https://www.entrepreneur.com/article/247467`.

Falk, R., & Konold, C. (1997). Making sense of randomness: Implicit encoding as a basis for judgment. *Psychol. Rev.*, *104*(2), 301.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Mach. Learn.*, *2*(2), 139–172.

Ganter, B., Obiedkov, S., Rudolph, S., & Stumme, G. (2016). *Conceptual Exploration*. Springer.

Ganter, B., & Wille, R. (2012). *Formal Concept Analysis: Mathematical Foundations*. Springer Science & Business Media.

Garg, V. K. (2015). *Introduction to Lattice Theory with Computer Science Applications*. Wiley Online Library.

Goertzel, B., & Pennachin, C. (2007). *Artificial General Intelligence*, Vol. 2. Springer.

Good, I. J. (1963). Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *Ann. Math. Stat.*, *34*(3), 911–934.

Hiller, L., & Isaacson, L. M. (1957). *Illiac Suite, for String Quartet*. New Music Edition.

Holzinger, A., Carrington, A., & Müller, H. (2020). Measuring the quality of explanations: the system causability scale (SCS). *Künstl. Intell.*, *34*(2), 193–198.

Holzinger, A., Malle, B., Saranti, A., & Pfeifer, B. (2021). Towards multi-modal causability with graph neural networks enabling information fusion for explainable AI. *Inf. Fusion*, *71*, 28–37.

Hutson, M. (2018). How researchers are teaching AI to learn like a child. `http://www.sciencemag.org/news/2018/05/how-researchers-are-teaching-ai-learn-child`.

Illiac Software, Inc. (2020). Harmonia. `https://harmonia.illiacsoftware.com/`.

Jain, A., Ong, S. P., Hautier, G., Chen, W., Richards, W. D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., & Persson, K. A. (2013). The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, *1*(1), 011002.

Jayasundara, V., Jayasekara, S., Jayasekara, H., Rajasegaran, J., Seneviratne, S., & Rodrigo, R. (2019). TextCaps: Handwritten character recognition with very small datasets. In *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, pp. 254–262.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *Phys. Rev.*, *106*(4), 620.

Jordan, M. I. (2019). Artificial intelligence—the revolution hasn't happened yet. *Harvard Data Science Review*, *1*(1).

Kaiser, D. (2005). Physics and Feynman's Diagrams: In the hands of a postwar generation, a tool intended to lead quantum electrodynamics out of a decades-long morass helped transform physics. *Am. Scientist*, *93*(2), 156–165.

Kauer, M., & Krupka, M. (2015). Subset-generated complete sublattices as concept lattices. In *Proc. 12th Int. Conf. Concept Lattices Appl.*, pp. 11–21.

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT press.

Kramer, M. A. (1991). Nonlinear principal component analysis using autoassociative neural networks. *AIChE J.*, *37*(2), 233–243.

Lipton, Z. C. (2018). The mythos of model interpretability. *Queue*, *16*(3), 31–57.

Liu, Z., & Tegmark, M. (2020). AI Poincaré: Machine learning conservation laws from trajectories. arXiv:2011.04698 [cs.LG].

Livingston, K. R. (1998). *Rationality and the Psychology of Abstraction*. Institute for Objectivist Studies.

MacNeille, H. M. (1937). Partially ordered sets. *Trans. Am. Math. Soc.*, *42*(3), 416–460.

Marcus, G. (2018). Innateness, AlphaZero, and artificial intelligence. arXiv:1801.05667 [cs.AI].

Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman.

Michalski, R. S., & Stepp, R. E. (1983). Learning from observation: Conceptual clustering. *Mach. Learn.*, *1*, 331–363.

Molnar, C. (2019). *Interpretable Machine Learning*. Lulu.com.

Pape, A. D., Kurtz, K. J., & Sayama, H. (2015). Complexity measures and concept learning. *J. Math. Psychol.*, *64*, 66–75.

Priss, U. (2006). Formal concept analysis in information science. *Ann. Rev. Inform. Sci. Tech.*, *40*(1), 521–543.

Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. arXiv:2002.12327 [cs.CL].

Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE Trans. Syst., Man, Cybern.*, *21*(3), 660–674.

Schmidt, M., & Lipson, H. (2009). Distilling free-form natural laws from experimental data. *Science*, *324*(5923), 81–85.

Selbst, A. D., & Barocas, S. (2018). The intuitive appeal of explainable machines. *Fordham L. Rev.*, *87*, 1085.

Shannon, C. (1953). The lattice theory of information. *Trans. IRE Prof. Group Inf. Theory*, *1*(1), 105–107.

Shin, M., Kim, J., van Opheusden, B., & Griffiths, T. L. (2023). Superhuman artificial intelligence can improve human decision-making by increasing novelty. *Proc. Natl. Acad. Sci. U.S.A.*, *120*(12), e2214840120.

Snow, C. P. (1959). *The Two Cultures*. Cambridge University Press.

Sokol, C. (2016). Figured soprano. `https://caseysokol.com/?page_id=1067`.

Spelke, E. S., & Kinzler, K. D. (2007). Core knowledge. *Developmental Sci.*, *10*(1), 89–96.

Thompson, D. (2014). Why experts reject creativity. `https://www.theatlantic.com/business/archive/2014/10/why-new-ideas-fail/381275/`.

Tymoczko, D. (2010). *A Geometry of Music: Harmony and Counterpoint in the Extended Common Practice*. Oxford University Press.

Udrescu, S.-M., & Tegmark, M. (2020). AI Feynman: A physics-inspired method for symbolic regression. *Sci. Adv.*, *6*(16), eaay2631.

Yu, H., Li, T., & Varshney, L. R. (2017). Probabilistic rule realization and selection. In *Proc. 31th Annu. Conf. Neural Inf. Process. Syst. (NeurIPS 2017)*, pp. 1561–1571.

Yu, H., & Varshney, L. R. (2017). Towards deep interpretability (MUS-ROVER II): Learning hierarchical representations of tonal music. In *Proc. 5th Int. Conf. Learn. Represent. (ICLR 2017)*.

Yu, H., Varshney, L. R., Garnett, G. E., & Kumar, R. (2016). MUS-ROVER: A self-learning system for musical compositional rules. In *Proc. 4th Int. Workshop Music. Metacreation (MUME 2016)*.

Zhou, Q., Tang, P., Liu, S., Pan, J., Yan, Q., & Zhang, S.-C. (2018). Learning atoms for materials discovery. *Proc. Natl. Acad. Sci. U.S.A.*, *115*(28), E6411–E6417.

Zhu, Y., Gao, T., Fan, L., Huang, S., Edmonds, M., Liu, H., Gao, F., Zhang, C., Qi, S., Wu, Y. N., Tenenbaum, J. B., & Zhu, S.-C. (2020). Dark, beyond deep: A paradigm shift to cognitive ai with humanlike common sense. *Engineering*, *6*(3), 310–345.