Interpretable Machine Learning: Fundamental Principles and 10 Grand Challenges

Cynthia Rudin[1], Chaofan Chen[2], Zhi Chen[1], Haiyang Huang[1], Lesia Semenova[1], and Chudi Zhong[1]

[1] Duke University
[2] University of Maine

July, 2021

Abstract

Interpretability in machine learning (ML) is crucial for high stakes decisions and troubleshooting. In this work, we provide fundamental principles for interpretable ML, and dispel common misunderstandings that dilute the importance of this crucial topic. We also identify 10 technical challenge areas in interpretable machine learning and provide history and background on each problem. Some of these problems are classically important, and some are recent problems that have arisen in the last few years. These problems are: (1) Optimizing sparse logical models such as decision trees; (2) Optimization of scoring systems; (3) Placing constraints into generalized additive models to encourage sparsity and better interpretability; (4) Modern case-based reasoning, including neural networks and matching for causal inference; (5) Complete supervised disentanglement of neural networks; (6) Complete or even partial unsupervised disentanglement of neural networks; (7) Dimensionality reduction for data visualization; (8) Machine learning models that can incorporate physics and other generative or causal constraints; (9) Characterization of the "Rashomon set" of good models; and (10) Interpretable reinforcement learning. This survey is suitable as a starting point for statisticians and computer scientists interested in working in interpretable machine learning.[1]

Introduction

With widespread use of machine learning (ML), the importance of interpretability has become clear in avoiding catastrophic consequences. Black box predictive models, which by definition are inscrutable, have led to serious societal problems that deeply affect health, freedom, racial bias, and safety. Interpretable predictive models, which are constrained so that their reasoning processes are more understandable to humans, are much easier to troubleshoot and to use in practice. It is universally agreed that interpretability is a key element of trust for AI models (Wagstaff, 2012; Rudin

[1]Equal contribution from C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong

and Wagstaff, 2014; Lo Piano, 2020; Ashoori and Weisz, 2019; Thiebes et al., 2020; Spiegelhalter, 2020; Brundage et al., 2020). In this survey, we provide fundamental principles, as well as 10 technical challenges in the design of inherently interpretable machine learning models.

Let us provide some background. A black box machine learning model is a formula that is either too complicated for any human to understand, or proprietary, so that one cannot understand its inner workings. Black box models are difficult to troubleshoot, which is particularly problematic for medical data. Black box models often predict the right answer for the wrong reason (the "Clever Hans" phenomenon), leading to excellent performance in training but poor performance in practice (Schramowski et al., 2020; Lapuschkin et al., 2019; O'Connor, 2021; Zech et al., 2018; Badgeley et al., 2019; Hamamoto et al., 2020). There are numerous other issues with black box models. In criminal justice, individuals may have been subjected to years of extra prison time due to typographical errors in black box model inputs (Wexler, 2017) and poorly-designed proprietary models for air quality have had serious consequences for public safety during wildfires (McGough, 2018); both of these situations may have been easy to avoid with interpretable models. In cases where the underlying distribution of data changes (called domain shift, which occurs often in practice), problems arise if users cannot troubleshoot the model in real-time, which is much harder with black box models than interpretable models. Determining whether a black box model is fair with respect to gender or racial groups is much more difficult than determining whether an interpretable model has such a bias. In medicine, black box models turn computer-aided decisions into automated decisions, precisely because physicians cannot understand the reasoning processes of black box models. Explaining black boxes, rather than replacing them with interpretable models, can make the problem worse by providing misleading or false characterizations (Rudin, 2019; Laugel et al., 2019; Lakkaraju and Bastani, 2020), or adding unnecessary authority to the black box (Rudin and Radin, 2019). There is a clear need for innovative machine learning models that are inherently interpretable.

There is now a vast and confusing literature on some combination of interpretability and explainability. Much literature on explainability confounds it with interpretability/comprehensibility, thus obscuring the arguments (and thus detracting from their precision), and failing to convey the relative importance and use-cases of the two topics in practice. Some of the literature discusses topics in such generality that its lessons have little bearing on any specific problem. Some of it aims to design taxonomies that miss vast topics within interpretable ML. Some of it provides definitions that we disagree with. Some of it even provides guidance that could perpetuate bad practice. Importantly, most of it assumes that one would explain a black box without consideration of whether there is an interpretable model of the same accuracy. In what follows, we provide some simple and general guiding principles of interpretable machine learning. These are not meant to be exhaustive. Instead they aim to help readers avoid common but problematic ways of thinking about interpretability in machine learning.

The major part of this survey outlines a set of important and fundamental technical grand challenges in interpretable machine learning. These are both modern and classical challenges, and some are much harder than others. They are all either hard to solve, or difficult to formulate correctly. While there are numerous sociotechnical challenges about model deployment (that can be much more difficult than technical challenges), human-computer-interaction challenges, and how robustness and fairness interact with interpretability, those topics can be saved for another day. We begin with the most classical and most canonical problems in interpretable machine learning: how to build sparse models for tabular data, including decision trees (Challenge #1) and scoring

systems (Challenge #2). We then delve into a challenge involving additive models (Challenge #3), followed by another in case-based reasoning (Challenge #4), which is another classic topic in interpretable artificial intelligence. We then move to more exotic problems, namely supervised and unsupervised disentanglement of concepts in neural networks (Challenges #5 and #6). Back to classical problems, we discuss dimension reduction (Challenge #7). Then, how to incorporate physics or causal constraints (Challenge #8). Challenge #9 involves understanding, exploring, and measuring the Rashomon set of accurate predictive models. Challenge #10 discusses interpretable reinforcement learning. Table 1 provides a guideline that may help users to match a dataset to a suitable interpretable supervised learning technique. We will touch on all of these techniques in the challenges.

Models	**Data type**
decision trees / decision lists (rule lists) / decision sets	somewhat clean tabular data with interactions, including multiclass problems. Particularly useful for categorical data with complex interactions (i.e., more than quadratic).
scoring systems	somewhat clean tabular data, typically used in medicine and criminal justice because they are small enough that they can be memorized by humans.
generalized additive models (GAMs)	continuous data with at most quadratic interactions, useful for large-scale medical record data.
case-based reasoning	any data type (different methods exist for different data types), including multiclass problems.
disentangled neural networks	data with raw inputs (computer vision, time series, textual data), suitable for multiclass problems.

Table 1: Rule of thumb for the types of data that naturally apply to various supervised learning algorithms. "Clean" means that the data do not have too much noise or systematic bias. "Tabular" means that the features are categorical or real, and that each feature is a meaningful predictor of the output on its own. "Raw" data is unprocessed and has a complex data type, e.g., image data where each pixel is a feature, medical records, or time series data.

General Principles of Interpretable Machine Learning

Our first fundamental principle defines interpretable ML, following Rudin (2019):

Principle 1 *An interpretable machine learning model obeys a domain-specific set of constraints to allow it (or its predictions, or the data) to be more easily understood by humans. These constraints can differ dramatically depending on the domain.*

A typical interpretable supervised learning setup, with data $\{z_i\}_i$, and models chosen from function class $\mathcal{F}$ is:

$$\min_{f\in\mathcal{F}} \frac{1}{n}\sum_i \text{Loss}(f, z_i) + C \cdot \text{InterpretabilityPenalty}(f), \quad \text{subject to} \quad \text{InterpretabilityConstraint}(f), \tag{1}$$

where the loss function, as well as soft and hard interpretability constraints, are chosen to match the domain. (For classification z_i might be (x_i, y_i), $x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}$.) The goal of these constraints is to make the resulting model f or its predictions more interpretable. While solutions of (1) would not necessarily be sufficiently interpretable to use in practice, the constraints would generally help us find models that would be interpretable (if we design them well), and we might also be willing to consider slightly suboptimal solutions to find a more useful model. The constant C trades off between accuracy and the interpretability penalty, and can be tuned, either by cross-validation or by taking into account the user's desired tradeoff between the two terms.

Equation (1) can be generalized to unsupervised learning, where the loss term is simply replaced with a loss term for the unsupervised problem, whether it is novelty detection, clustering, dimension reduction, or another task.

Creating interpretable models can sometimes be much more difficult than creating black box models for many different reasons including: (i) Solving the optimization problem may be computationally hard, depending on the choice of constraints and the model class $\mathcal{F}$. (ii) When one does create an interpretable model, one invariably realizes that the data are problematic and require troubleshooting, which slows down deployment (but leads to a better model). (iii) It might not be initially clear which definition of interpretability to use. This definition might require refinement, sometimes over multiple iterations with domain experts. There are many papers detailing these issues, the earliest dating from the mid-1990s (e.g., Kodratoff, 1994).

Interpretability differs across domains just as predictive performance metrics vary across domains. Just as we might choose from a variety of performance metrics (e.g., accuracy, weighted accuracy, precision, average precision, precision@N, recall, recall@N, DCG, NCDG, AUC, partial AUC, mean-time-to-failure, etc.), or combinations of these metrics, we might also choose from a combination of interpretability metrics that are specific to the domain. We may not be able to define a single best definition of interpretability; regardless, if our chosen interpretability measure is helpful for the problem at hand, we are better off including it. Interpretability penalties or constraints can include sparsity of the model, monotonicity with respect to a variable, decomposibility into sub-models, an ability to perform case-based reasoning or other types of visual comparisons, disentanglement of certain types of information within the model's reasoning process, generative constraints (e.g., laws of physics), preferences among the choice of variables, or any other type of constraint that is relevant to the domain. Just as it would be futile to create a complete list of performance metrics for machine learning, any list of interpretability metrics would be similarly fated.

Our 10 challenges involve how to define some of these interpretability constraints and how to incorporate them into machine learning models. For tabular data, sparsity is usually part of the definition of interpretability, whereas for computer vision of natural images, it generally is not. (Would you find a model for natural image classification interpretable if it uses only a few pixels from the image?) For natural images, we are better off with interpretable neural networks that perform case-based reasoning or disentanglement, and provide us with a visual understanding of intermediate computations; we will describe these in depth. Choices of model form (e.g., the choice to use a decision tree, or a specific neural architecture) are examples of interpretability constraints. For most problems involving tabular data, a fully interpretable model, whose full calculations can be understood by humans such as a sparse decision tree or sparse linear model, is generally more desirable than either a model whose calculations can only be partially understood, or a model whose predictions (but not its model) can be understood. Thus, we make a distinction

between *fully interpretable* and *partially interpretable* models, often preferring the former.

Interpretable machine learning models are not needed for all machine learning problems. For low-stakes decisions (e.g., advertising), for decisions where an explanation would be trivial and the model is 100% reliable (e.g., "there is no lesion in this mammogram" where the explanation would be trivial), for decisions where humans can verify or modify the decision afterwards (e.g., segmentation of the chambers of the heart), interpretability is probably not needed.[2] On the other hand, for self-driving cars, even if they are very reliable, problems would arise if the car's vision system malfunctions causing a crash and no reason for the crash is available. Lack of interpretability would be problematic in this case.

Our second fundamental principle concerns trust:

Principle 2 *Despite common rhetoric, interpretable models do not necessarily create or enable trust – they could also enable* dis*trust. They simply allow users to* decide *whether to trust them. In other words, they permit a decision of trust, rather than trust itself.*

With black boxes, one needs to make a decision about trust with much less information; without knowledge about the reasoning process of the model, it is much more difficult to detect whether it might generalize beyond the dataset. As stated by Afnan et al. (2021) with respect to medical decisions, while interpretable AI is an enhancement of human decision making, black box AI is a replacement of it.

An important point about interpretable machine learning models is that *there is no scientific evidence for a general tradeoff between accuracy and interpretability* when one considers the full data science process for turning data into knowledge. (Examples of such pipelines include KDD, CRISP-DM, or the CCC Big Data Pipelines; see Figure 1, or Fayyad et al. 1996; Chapman et al. 2000; Agrawal et al. 2012.) In real problems, interpretability is useful for troubleshooting, which

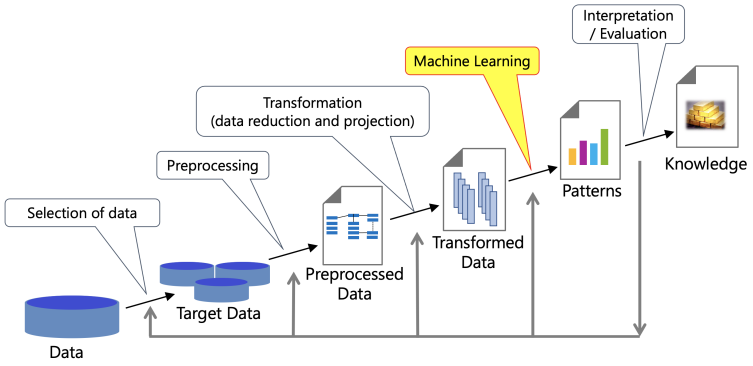


Figure 1: Knowledge Discovery Process (figure adapted from Fayyad et al., 1996)

.

leads to better accuracy, not worse. In that sense, we have the third principle:

Principle 3 *It is important not to assume that one needs to make a sacrifice in accuracy in order to gain interpretability. In fact, interpretability often begets accuracy, and not the reverse. Inter-*

[2]Obviously, this document does not apply to black box formulas that not depend on randomness in the data, i.e., a calculation of deterministic function, not machine learning.

pretability versus accuracy is, in general, a false dichotomy in machine learning.

Interpretability has traditionally been associated with complexity, and specifically, sparsity, but model creators generally would not *equate* interpretability with sparsity. Sparsity is often one component of interpretability, and a model that is sufficiently sparse but has other desirable properties is more typical. While there is almost always a tradeoff of accuracy with sparsity (particularly for extremely small models), there is no evidence of a general tradeoff of accuracy with interpretability. Let us consider both (1) development and use of ML models in practice, and (2) experiments with static datasets; in neither case have interpretable models proven to be less accurate.

Development and use of ML models in practice. A key example of the pitfalls of black box models when used in practice is the story of COMPAS (Northpointe, 2013; Rudin et al., 2020). COMPAS is a black box because it is proprietary: no one outside of its designers knows its secret formula for predicting criminal recidivism, yet it is used widely across the United States and influences parole, bail, and sentencing decisions that deeply affect people's lives (Angwin et al., 2016). COMPAS is *error-prone* because it could require over 130 variables, and typographical errors in those variables influence outcomes (Wexler, 2017). COMPAS has been *explained incorrectly*, where the news organization ProPublica mistakenly assumed that an important variable in an approximation to COMPAS (namely race) was also important to COMPAS itself, and used this faulty logic to conclude that COMPAS depends on race other than through age and criminal history (Angwin et al., 2016; Rudin et al., 2020). While the racial bias of COMPAS does not seem to be what ProPublica claimed, COMPAS still has *unclear dependence on race*. And worst, COMPAS seems to be *unnecessarily complicated* as it does not seem to be any more accurate than a very sparse decision tree (Angelino et al., 2017, 2018) involving only a couple of variables. It is a key example simultaneously demonstrating many pitfalls of black boxes in practice. And, it shows an example of when black boxes are unnecessary, but used anyway.

There are other systemic issues that arise when using black box models in practice or when developing them as part of a data science process, that would cause them to be less accurate than an interpretable model. A full list is beyond the scope of this particular survey, but an article that points out many serious issues that arise with black box models in a particular domain is that of Afnan et al. (2021), who discuss in vitro fertilization (IVF). In modern IVF, black box models that have not been subjected to randomized controlled trials determine who comes into existence. Afnan et al. (2021) points out ethical and practical issues, ranging from the inability to perform shared decision-making with patients, to economic consequences of clinics needing to "buy into" environments that are similar to those where the black box models are trained so as to avoid distribution shift; this is necessary precisely because errors cannot be detected effectively in real-time using the black box. They also discuss accountability issues ("Who is accountable when the model causes harm?"). Finally, they present a case where a standard performance metric (namely area under the ROC curve – AUC) has been misconstrued as representing the value of a model in practice, potentially leading to overconfidence in the performance of a black box model. Specifically, a reported AUC was inflated by including many "obvious" cases in the sample over which it was computed. If we cannot trust reported numerical performance results, then interpretability would be a crucial remaining ingredient to assess trust.

In a full data science process, like the one shown in Figure 1, interpretability plays a key role in determining how to update the other steps of the process for the next iteration. One interprets the

results and tunes the processing of the data, the loss function, the evaluation metric, or anything else that is relevant, as shown in the figure. How can one do this without understanding how the model works? It may be possible, but might be much more difficult. In essence, the messiness that comes with messy data and complicated black box models causes lower quality decision-making in practice.

Let's move on to a case where the problem is instead controlled, so that we have a static dataset and a fixed evaluation metric.

Static datasets. Most benchmarking of algorithms is done on static datasets, where the data and evaluation metric are not cleaned or updated as a result of a run of an algorithm. In other words, these experiments are not done as part of a data science process, they are only designed to compare algorithms in a controlled experimental environment.

Even with static datasets and fixed evaluation metrics, interpretable models do not generally lead to a loss in accuracy over black box models. Even for deep neural networks for computer vision, even on the most challenging of benchmark datasets, a plethora of machine learning techniques have been designed *that do not sacrifice accuracy but gain substantial interpretability* (Zhang et al., 2018; Chen et al., 2019; Koh et al., 2020; Chen et al., 2020; Angelov and Soares, 2020; Nauta et al., 2020b).

Let us consider two extremes of data types: tabular data, where all variables are real or discrete features, each of which is meaningful (e.g., age, race, sex, number of past strokes, congestive heart failure), and "raw" data, such as images, sound files or text, where each pixel, bit, or word is not useful on its own. These types of data have different properties with respect to both machine learning performance and interpretability.

For tabular data, most machine learning algorithms tend to perform similarly in terms of prediction accuracy. This means it is often difficult even to beat logistic regression, assuming one is willing to perform minor preprocessing such as creating dummy variables (e.g., see Christodoulou et al., 2019). In these domains, neural networks generally find no advantage. It has been known for a very long time that very simple models perform surprisingly well for tabular data (Holte, 1993). The fact that simple models perform well for tabular data could arise from the Rashomon Effect discussed by Leo Breiman (Breiman et al., 2001). Breiman posits the possibility of a large Rashomon set, i.e., a multitude of models with approximately the minumum error rate, for many problems. Semenova et al. (2019) show that as long as a large Rashomon set exists, it is more likely that some of these models are interpretable.

For raw data, on the other hand, neural networks have an advantage currently over other approaches (Krizhevsky et al., 2017). In these raw data cases, the definition of interpretability changes; for visual data, one may require visual explanations. In such cases, as discussed earlier and in Challenges 4 and 5, interpretable neural networks suffice, without losing accuracy.

These two data extremes show that in machine learning, the dichotomy between the accurate black box and the less-accurate interpretable model is false. The often-discussed hypothetical choice between the accurate machine-learning-based robotic surgeon and the less-accurate human surgeon is moot once someone builds an interpretable robotic surgeon. Given that even the most difficult computer vision benchmarks can be solved with interpretable models, there is no reason to believe that an interpretable robotic surgeon would be worse than its black box counterpart. The question ultimately becomes whether the Rashomon set should permit such an interpretable robotic surgeon–and all scientific evidence so far (including a large-and-growing number of experimental

papers on interpretable deep learning) suggests it would.

Our next principle returns to the data science process.

Principle 4 *As part of the full data science process, one should expect both the performance metric and interpretability metric to be iteratively refined.*

The knowledge discovery process in Figure 1 explicitly shows these important feedback loops. We have found it useful in practice to create many interpretable models (satisfying the known constraints) and have domain experts choose between them. Their rationale for choosing one model over another helps to refine the definition of interpretability. Each problem can thus have its own unique interpretability metrics (or set of metrics).

The fifth principle is as follows:

Principle 5 *For high stakes decisions, interpretable models should be used if possible, rather than "explained" black box models.*

Hence, this survey concerns the former. This is not a survey on Explainable AI (XAI, where one attempts to explain a black box using an approximation model, derivatives, variable importance measures, or other statistics), it is a survey on *Interpretable Machine Learning* (creating a predictive model that is not a black box). Unfortunately, these topics are much too often lumped together within the misleading term "explainable artificial intelligence" or "XAI" despite a chasm separating these two concepts (Rudin, 2019). Explainability and interpretability techniques are not alternative choices for many real problems, as the recent surveys often imply; one of them (XAI) can be dangerous for high-stakes decisions to a degree that the other is not.

Interpretable ML is not a subset of XAI. The term XAI dates from $\sim$2016, and grew out of work on function approximation; i.e., explaining a black box model by approximating its predictions by a simpler model (e.g., Craven and Shavlik, 1995; Craven, 1996), or explaining a black box using local approximations. Interpretable ML also has a (separate) long and rich history, dating back to the days of expert systems in the 1950's, and the early days of decision trees. While these topics may sound similar to some readers, they differ in ways that are important in practice.

In particular, there are many serious problems with the use of explaining black boxes posthoc, as outlined in several papers that have shown why explaining black boxes can be misleading and why explanations do not generally serve their intended purpose (Rudin, 2019; Laugel et al., 2019; Lakkaraju and Bastani, 2020). The most compelling such reasons are:

- Explanations for black boxes are often problematic and misleading, potentially creating misplaced trust in black box models. Such issues with explanations have arisen with assessment of fairness and variable importance (Rudin et al., 2020; Dimanov et al., 2020) as well as uncertainty bands for variable importance (Gosiewska and Biecek, 2020; Fisher et al., 2019). There is an overall difficulty in troubleshooting the combination of a black box and an explanation model on top of it; if the explanation model is not always correct, it can be difficult to tell whether the black box model is wrong, or if it is right and the explanation model is wrong. Ultimately, posthoc explanations are wrong (or misleading) too often.

  One particular type of posthoc explanation, called saliency maps (also called attention maps) have become particularly popular in radiology and other computer vision domains despite

known problems (Adebayo et al., 2018; Chen et al., 2019; Zhang et al., 2019). Saliency maps highlight the pixels of an image that are used for a prediction, but they do not explain how the pixels are used. As an analogy, consider a real estate agent who is pricing a house. A "black box" real estate agent would provide the price with no explanation. A "saliency" real estate agent would say that the price is determined from the roof and backyard, but doesn't explain how the roof and backyard were used to determine the price. In contrast, an interpretable agent would explain the calculation in detail, for instance, using "comps" or comparable properties to explain how the roof and backyard are comparable between properties, and how these comparisons were used to determine the price. One can see from this real estate example how the saliency agent's explanation is insufficient.

Saliency maps also tend to be unreliable; researchers often report that different saliency methods provide different results, making it unclear which one (if any) actually represents the network's true attention.[3]

- Black boxes are generally unnecessary, given that their accuracy is generally not better than a well-designed interpretable model. Thus, explanations that seem reasonable can undermine efforts to find an interpretable model of the same level of accuracy as the black box.

- Explanations for complex models hide the fact that complex models are difficult to use in practice for many different reasons. Typographical errors in input data are a prime example of this issue (as in the use of COMPAS in practice, see Wexler, 2017). A model with 130 hand-typed inputs is more error-prone than one involving 5 hand-typed inputs.

In that sense, *explainability methods are often used as an excuse to use a black box model–whether or not one is actually needed.* Explainability techniques give authority to black box models rather than suggesting the possibility of models that are understandable in the first place (Rudin and Radin, 2019).

XAI surveys have (thus far) universally failed to acknowledge the important point that interpretability begets accuracy when considering the full data science process, and not the other way around. Perhaps this point is missed because of the more subtle fact that one does generally lose accuracy when approximating a complicated function with a simpler one, and these imperfect approximations are the foundation of XAI. (Again the approximations must be imperfect, otherwise one would throw out the black box and instead use the explanation as an inherently interpretable model.) But function approximators are not used in interpretable ML; instead of approximating a known function (a black box ML model), interpretable ML can choose from a potential myriad of approximately-equally-good models, which, as we noted earlier, is called "the Rashomon set" (Breiman et al., 2001; Fisher et al., 2019; Semenova et al., 2019). We will discuss the study of this

[3]To clear possible confusion, techniques such as SHAP and LIME are tools for explaining black box models, are not needed for inherently interpretable models, and thus do not belong in this survey. These methods determine how much each variable contributed to a prediction. Interpretable models do not need SHAP values because they already explicitly show what variables they are using and how they are using them. For instance, sparse decision trees and sparse linear models do not need SHAP values because we know exactly what variables are being used and how they are used. Interpretable supervised deep neural networks that use case-based reasoning (Challenge #4) do not need SHAP values because they explicitly reveal what part of the observation they are paying attention to, and in addition, how that information is being used (e.g., what comparison is being made between part of a test image and part of a training image). Thus, if one creates an interpretable model, one does not need LIME or SHAP whatsoever.

set in Challenge 9. Thus, *when one explains black boxes, one expects to lose accuracy, whereas when one creates an inherently interpretable ML model, one does not.*

In this survey, we do not aim to provide yet another dull taxonomy of "explainability" terminology. The ideas of interpretable ML can be stated in just one sentence: an interpretable model is constrained, following a domain-specific set of constraints that make reasoning processes understandable. Instead, we highlight important challenges, each of which can serve as a starting point for someone wanting to enter into the field of interpretable ML.

1 Sparse Logical Models: Decision Trees, Decision Lists, and Decision Sets

The first two challenges involve optimization of sparse models. We discuss both sparse logical models in Challenge #1 and scoring systems (which are sparse linear models with integer coefficients) in Challenge #2. Sparsity is often used as a measure of interpretability for tabular data where the features are meaningful. Sparsity is useful because humans can handle only 7 ± 2 cognitive entities at the same time (Miller, 1956), and sparsity makes it easier to troubleshoot, check for typographical errors, and reason about counterfactuals (e.g., "How would my prediction change if I changed this specific input?"). Sparsity is rarely the only consideration for interpretability, but if we can design models to be sparse, we can often handle additional constraints. Also, if one can optimize for sparsity, a useful baseline can be established for how sparse a model could be with a particular level of accuracy.

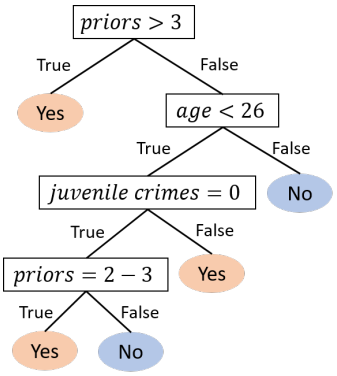
We remark that more sparsity does not always equate to more interpretability. Elomaa (1994) and Freitas (2014) make the point that "Humans by nature are mentally opposed to too simplistic representations of complex relations." For instance, in loan decisions, we may choose to have several sparse mini-models for length of credit, history of default, etc., which are then assembled at the end into a larger model composed of the results of the mini-models (see Chen et al., 2021a, who attempted this). On the other hand, sparsity is necessary for many real applications, particularly in healthcare and criminal justice where the practitioner needs to memorize the model.

Logical models, which consist of logical statements involving "if-then," "or," and "and" clauses are among the most popular algorithms for interpretable machine learning, since their statements provide human-understandable reasons for each prediction.

When would we use logical models? Logical models are usually an excellent choice for modeling categorical data with potentially complicated interaction terms (e.g., "IF (female AND high blood pressure AND congenital heart failure), OR (male AND high blood pressure AND either prior stroke OR age $>$ 70) THEN predict Condition 1 = true"). Logical models are also excellent for multiclass problems. Logical models are also known for their robustness to outliers and ease of handling missing data. Logical models can be highly nonlinear, and even classes of sparse nonlinear models can be quite powerful.

Figure 2 visualizes three logical models: a decision tree, a decision list, and a decision set. *Decision trees* are tree-structured predictive models where each branch node tests a condition and each leaf node makes a prediction. *Decision lists*, identical to rule lists or one-sided decision trees, are composed of if-then-else statements. The rules are tried in order, and the first rule that is satisfied makes the prediction. Sometimes rule lists have multiple conditions in each split, whereas

decision trees typically do not. A *decision set*, also known as a "disjunction of conjunctions," "disjunctive normal form" (DNF), or an "OR of ANDs" is comprised of an unordered collection of rules, where each rule is a conjunction of conditions. A positive prediction is made if at least one of the rules is satisfied. Even though these logical models seem to have very different forms, they are closely related: every decision list is a (one-sided) decision tree and every decision tree can be expressed as an equivalent decision list (by listing each path to a leaf as a decision rule). The collection of leaves of a decision tree (or a decision list) also forms a decision set.



(a)

if ($age < 26$) **and** ($priors = 2-3$) **then predict** yes
else if ($juvenile\ crimes = 0$) **and** ($priors < 3$) **then predict** no
else predict yes

(b)

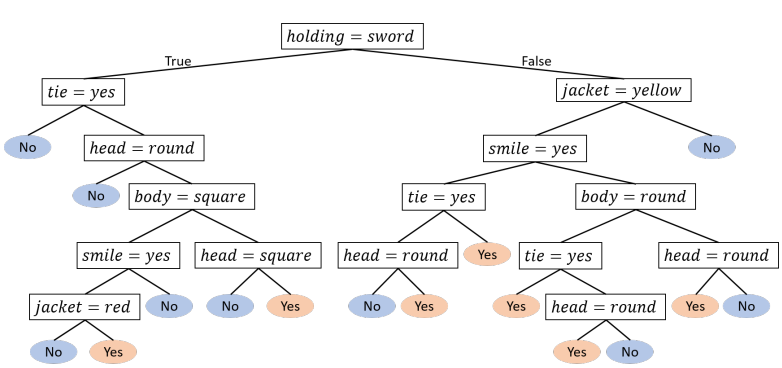
if ($priors > 3$) **and** ($age < 21$) **then predict** yes
if ($juvenile\ crimes > 0$) **and** ($prior > 3$) **then predict** yes
if ($age < 23$) **and** ($prior = 2-3$) **then predict** yes
else predict no

(c)

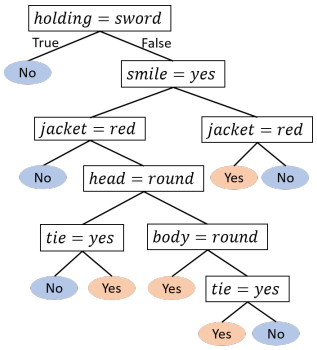
Figure 2: Predicting which individuals are arrested within two years of release by a decision tree (a), a decision list (b), and a decision set (c). The dataset used here is the ProPublica recidivism dataset (Angwin et al., 2016)

Let us provide some background on decision trees. Since Morgan and Sonquist (1963) developed the first decision tree algorithm, many works have been proposed to build decision trees and improve their performance. However, learning decision trees with high performance and sparsity is not easy. Full decision tree optimization is known to be an NP-complete problem (Laurent and Rivest, 1976), and heuristic greedy splitting and pruning procedures have been the major type of approach since the 1980s to grow decision trees (Breiman et al., 1984; Quinlan, 1993; Loh and Shih, 1997; Mehta et al., 1996). These greedy methods for building decision trees create trees from the top down and prune them back afterwards. They do not go back to fix a bad split if one was made. Consequently, the trees created from these greedy methods tend to be both less accurate and less interpretable than necessary. That is, greedy induction algorithms are not designed to optimize any particular performance metric, leaving a gap between the performance that a decision tree might obtain and the performance that the algorithm's decision tree actually attains, with no way to determine how large the gap is (see Figure 3 for a case where CART did not obtain an optimal solution, as shown by the better solution from a more recent algorithm called "GOSDT," to the right). This gap can cause a problem in practice because one does not know whether poor performance is due to the choice of model form (the choice to use a decision tree of a specific size) or poor optimization (not fully optimizing over the set of decision trees of that size). When fully optimized, single trees can be as accurate as ensembles of trees, or neural networks, for many problems. Thus, it is worthwhile to think carefully about how to optimize them.

GOSDT and related modern decision tree methods solve an optimization problem that is a



(a) training accuracy: 75.74%; test accuracy: 69.44%



(b) training accuracy: 81.07%; test accuracy: 73.15%

Figure 3: (a) 16-leaf decision tree learned by CART (Breiman et al., 1984) and (b) 9-leaf decision tree generated by GOSDT (Lin et al., 2020) for the classic Monk 2 dataset (Dua and Graff, 2017). The GOSDT tree is optimal with respect to a balance between accuracy and sparsity.

special case of (1):

$$\min_{f \in \text{ set of trees}} \frac{1}{n} \sum_i \text{Loss}(f, z_i) + C \cdot \text{Number of leaves } (f), \tag{2}$$

where the user specifies the loss function and the trade-off (regularization) parameter.

Efforts to fully optimize decision trees, solving problems related to (2), have been made since the 1990s (Bennett and Blue, 1996; Dobkin et al., 1997; Farhangfar et al., 2008; Nijssen and Fromont, 2007, 2010; Hu et al., 2019; Lin et al., 2020). Many recent papers directly optimize the performance metric (e.g., accuracy) with soft or hard sparsity constraints on the tree size, where sparsity is measured by the number of leaves in the tree. Three major groups of these techniques are (1) mathematical programming, including mixed integer programming (MIP) (see the works of Bennett and Blue, 1996; Rudin and Ertekin, 2018; Verwer and Zhang, 2019; Vilas Boas et al., 2019; Menickelly et al., 2018; Aghaei et al., 2020) and SAT solvers (Narodytska et al., 2018; Hu et al., 2020) (see also the review of Carrizosa et al., 2021), (2) stochastic search through the space of trees (e.g., Yang et al., 2017; Gray and Fan, 2008; Papagelis and Kalles, 2000), and (3) customized dynamic programming algorithms that incorporate branch-and-bound techniques for reducing the size of the search space (Hu et al., 2019; Lin et al., 2020; Nijssen et al., 2020; Demirović et al., 2020).

Decision list and decision set construction lead to the same challenges as decision tree optimization, and have a parallel development path. Dating back to 1980s, decision lists have often been constructed in a top-down greedy fashion. Associative classification methods assemble decision lists or decision sets from a set of pre-mined rules, generally either by greedily adding rules to the model one by one, or simply including all "top-scoring" rules into a decision set, where each rule is scored separately according to a scoring function (Rivest, 1987; Clark and Niblett, 1989; Liu et al., 1998; Li et al., 2001; Yin and Han, 2003; Sokolova et al., 2003; Marchand and Sokolova, 2005; Vanhoof and Depaire, 2010; Rudin et al., 2013; Clark and Boswell, 1991; Gaines and Compton, 1995; Cohen, 1995; Frank and Witten, 1998; Friedman and Fisher, 1999; Marchand and Shawe-Taylor, 2002; Malioutov and Varshney, 2013). Sometimes decision lists or decision

sets are optimized by sampling (Letham et al., 2015; Yang et al., 2017; Wang and Rudin, 2015a), providing a Bayesian interpretation. Some recent works can jointly optimize performance metrics and sparsity for decision lists (Rudin and Ertekin, 2018; Yu et al., 2020a; Angelino et al., 2017, 2018; Aïvodji et al., 2019) and decision sets (Wang and Rudin, 2015b; Goh and Rudin, 2014; Lakkaraju et al., 2016; Ignatiev et al., 2018; Dash et al., 2018; Malioutov and Meel, 2018; Ghosh and Meel, 2019; Dhamnani et al., 2019; Yu et al., 2020b; Cao et al., 2020). Some works optimize for individual rules (Dash et al., 2018; Rudin and Shaposhnik, 2019)

Over the last few years, great progress has been made on optimizing the combination of accuracy and sparsity for logical models, but there are still many challenges that need to be solved. Some of the most important ones are as follows:

1.1 **Can we improve the scalability of optimal sparse decision trees?** A lofty goal for optimal decision tree methods is to fully optimize trees as fast as CART produces its (non-optimal) trees. Current state-of-the-art optimal decision tree methods can handle medium-sized datasets (thousands of samples, tens of binary variables) in a reasonable amount of time (e.g., within 10 minutes) when appropriate sparsity constraints are used. But how to scale up to deal with large datasets or to reduce the running time remains a challenge.

These methods often scale exponentially in p, the number of dimensions of the data. Developing algorithms that reduce the number of dimensions through variable screening theorems or through other means could be extremely helpful.

For methods that use the mathematical programming solvers, a good formulation is key to reducing training time. For example, MIP solvers use branch-and-bound methods, which partition the search space recursively and solve Linear Programming (LP) relaxations for each partition to produce lower bounds. Small formulations with fewer variables and constraints can enable the LP relaxations to be solved faster, while stronger LP relaxations (which usually involve more variables) can produce high quality lower bounds to prune the search space faster and reduce the number of LPs to be solved. How to formulate the problem to leverage the full power of MIP solvers is an open question. Currently, mathematical programming solvers are not as efficient as the best customized algorithms.

For customized branch-and-bound search algorithms such as GOSDT and OSDT (Lin et al., 2020; Hu et al., 2019), there are several mechanisms to improve scalability: (1) effective lower bounds, which prevent branching into parts of the search space where we can prove there is no optimal solution, (2) effective scheduling policies, which help us search the space to find close-to-optimal solutions quickly, which in turn improves the bounds and again prevents us from branching into irrelevant parts of the space, (3) computational reuse, whereby if a computation involves a sum over (even slightly) expensive computations, and part of that sum has previously been computed and stored, we can reuse the previous computation instead of computing the whole sum over again, (4) efficient data structures to store subproblems that can be referenced later in the computation should that subproblem arise again.

1.2 **Can we efficiently handle continuous variables?** While decision trees handle categorical variables and complicated interactions better than other types of approaches (e.g., linear models), one of the most important challenges for decision tree algorithms is to optimize over continuous features. Many current methods use binary variables as input (Hu et al., 2019; Lin

et al., 2020; Verwer and Zhang, 2019; Nijssen et al., 2020; Demirović et al., 2020), which assumes that continuous variables have been transformed into indicator variables beforehand (e.g., age> 50). These methods are unable to jointly optimize the selection of variables to split at each internal tree node, the splitting threshold of that variable (if it is continuous), and the tree structure (the overall shape of the tree). Lin et al. (2020) preprocesses the data by transforming continuous features into a set of dummy variables, with many different split points; they take split points at the mean values between every ordered pair of unique values present in the training data. Doing this preserves optimality, but creates a huge number of binary features, leading to a dramatic increase in the size of the search space, and the possibility of hitting either time or memory limits. Verwer and Zhang (2019); Nijssen et al. (2020) preprocess the data using an approximation, whereby they consider a much smaller subset of all possible thresholds, potentially sacrificing the optimality of the solution (see Lin et al., 2020, Section 3, which explains this). One possible technique to help with this problem is to use *similar support* bounds, identified by (Angelino et al., 2018), but in practice these bounds have been hard to implement because checking the bounds repeatedly is computationally expensive, to the point where the bounds have never been used (as far as we know). Future work could go into improving the determination of when to check these bounds, or proving that a subset of all possible dummy variables still preserves closeness to optimality.

1.3 **Can we handle constraints more gracefully?** Particularly for greedy methods that create trees using local decisions, it is difficult to enforce global constraints on the overall tree. Given that domain-specific constraints may be essential for interpretability, an important challenge is to determine how to incorporate such constraints. Optimization approaches (mathematical programming, dynamic programming, branch-and-bound) are more amenable to global constraints, but the constraints can make the problem much more difficult. For instance, *falling constraints* (Wang and Rudin, 2015a; Chen and Rudin, 2018) enforce decreasing probabilities along a rule list, which make the list more interpretable and useful in practice, but make the optimization problem harder, even though the search space itself becomes smaller.

Example: Suppose a hospital would like to create a decision tree that will be used to assign medical treatments to patients. A tree is convenient because it corresponds to a set of questions to ask the patient (one for each internal node along a path to a leaf). A tree is also convenient in its handling of multiple medical treatments; each leaf could even represent a different medical treatment. The tree can also handle complex interactions, where patients can be asked multiple questions that build on each other to determine the best medication for the patient. To train this tree, the proper assumptions and data handling were made to allow us to use machine learning to perform causal analysis (in practice these are more difficult than we have room to discuss here). The questions we discussed above arise when the variables are continuous; for instance, if we split on age somewhere in the tree, what is the optimal age to split at in order to create a sparse tree? (See Challenge 1.2.) If we have many other continuous variables (e.g., blood pressure, weight, body mass index), scalability in how to determine how to split them all becomes an issue. Further, if the hospital has additional preferences, such as "falling probabilities," where fewer questions should be asked to determine whether a patient is in the most urgent treatment categories, again it could affect our ability to find an optimal tree given limited computational resources (see Challenge 1.3).

2 Scoring systems

Scoring systems are linear classification models that require users to add, subtract, and multiply only a few small numbers in order to make a prediction. These models are used to assess the risk of numerous serious medical conditions since they allow quick predictions, without the use of a computer. Such models are also heavily used in criminal justice. Table 2 shows an example of a scoring system. A doctor can easily determine whether a patient screens positive for obstructive sleep apnea by adding points for the patient's age, whether they have diabetes, body mass index, and sex. If the score is above a threshold, the patient would be recommended to a clinic for a diagnostic test.

Scoring systems commonly use binary or indicator variables (e.g., age ≥ 60) and point values (e.g., Table 2) which make computation of the score easier for humans. Linear models do not handle interaction terms between variables like the logical models discussed above in Challenge 1.1, and they are not particularly useful for multiclass problems, but they are useful for counterfactual reasoning: if we ask "What changes could keep someone's score low if they developed hypertension?" the answer would be easy to compute using point scores such as "4, 4, 2, 2, and -6." Such logic is more difficult for humans if the point scores are instead, for instance, 53.2, 41.1, 16.3, 23.6 and -61.8.

Patient screens positive for obstructive sleep apnea if Score >1			
1.	age ≥ 60	4 points	.....
2.	hypertension	4 points	$+$ .....
3.	body mass index ≥ 30	2 points	$+$ .....
4.	body mass index ≥ 40	2 points	$+$ .....
5.	female	-6 points	$+$ .....
	Add points from row 1-6	Score	$=$ .....

Table 2: A scoring system for sleep apnea screening (Ustun et al., 2016). Patients that screen positive may need to come to the clinic to be tested.

Risk scores are scoring systems that have a conversion table to probabilities. For instance, a 1 point total might convert to probability 15%, 2 points to 33% and so on. Whereas scoring systems with a threshold (like the one in Table 2) would be measured by false positive and false negative rates, a risk score might be measured using the area under the ROC curve (AUC) and calibration.

The development of scoring systems dates back at least to criminal justice work in the 1920s (Burgess, 1928). Since then, many scoring systems have been designed for healthcare (Knaus et al., 1981, 1985, 1991; Bone et al., 1992; Le Gall et al., 1993; Antman et al., 2000; Gage et al., 2001; Moreno et al., 2005; Six et al., 2008; Weathers et al., 2013; Than et al., 2014). However, none of the scoring systems mentioned so far was optimized purely using an algorithm applied to data. Each scoring system was created using a different method involving different heuristics. Some of them were built using domain expertise alone without data, and some were created using rounding heuristics for logistic regression coefficients and other manual feature selection approaches to obtain integer-valued point scores (see, e.g., Le Gall et al., 1993).

Such scoring systems could be optimized using a combination of the user's preferences (and constraints) and data. This optimization should ideally be accomplished by a computer, leaving

the domain expert only to specify the problem. However, jointly optimizing for predictive performance, sparsity, and other user constraints may not be an easy computational task. Equation (3) shows an example of a generic optimization problem for creating a scoring system:

$$\min_{f\in\mathcal{F}} \quad \frac{1}{n}\sum_i \text{Loss}(f, z_i) + C \cdot \text{Number of nonzero terms } (f), \quad \text{subject to} \tag{3}$$

$$f \text{ is a linear model, } f(\mathbf{x}) = \sum_{j=1}^{p} \lambda_j x_j,$$

$$\text{with small integer coefficients, that is, } \forall\ j,\ \lambda_j \in \{-10, -9, .., 0, .., 9, 10\}$$

$$\text{and additional user constraints.}$$

Ideally, the user would specify the loss function (classification loss, logistic loss, etc.), the tradeoff parameter C between the number of nonzero coefficients and the training loss, and possibly some additional constraints, depending on the domain.

The integrality constraint on the coefficients makes the optimization problem very difficult. The easiest way to satisfy these constraints is to fit real coefficients (e.g., run logistic regression, perhaps with ℓ_1 regularization) and then round these real coefficients to integers. However, rounding can go against the loss gradient and ruin predictive performance. Here is an example of a coefficient vector that illustrates why rounding might not work:

$$[5.3, 6.1, 0.31, 0.30, 0.25, 0.25, 0.24, ..., 0.05] \quad \rightarrow \text{(rounding)} \rightarrow \quad [5, 6, 0, 0, ..., 0].$$

When rounding, we lose all signal coming from all variables except the first two. The contribution from the eliminated variables may together be significant even if each individual coefficient is small, in which case, we lose predictive performance.

Compounding the issue with rounding is the fact that ℓ_1 regularization introduces a strong bias for very sparse problems. To understand why, consider that the regularization parameter must be set to a very large number to get a very sparse solution. In that case, the ℓ_1 regularization does more than make the solution sparse, it also imposes a strong ℓ_1 bias. The solutions disintegrate in quality as the solutions become sparser, then rounding to integers only makes the solution worse.

An even bigger problem arises when trying to incorporate additional constraints, as we allude to in (3). Even simple constraints such as "ensure precision is at least 20%" when optimizing recall would be very difficult to satisfy manually with rounding. There are four main types of approaches to building scoring systems: i) exact solutions using optimization techniques, ii) approximation algorithms using linear programming, iii) more sophisticated rounding techniques, iv) computer-aided exploration techniques.

Exact solutions. There are several methods that can solve (3) directly (Ustun et al., 2013; Ustun and Rudin, 2016, 2017, 2019; Rudin and Ustun, 2018). To date, the most promising approaches use mixed-integer linear programming solvers (MIP solvers) which are generic optimization software packages that handle systems of linear equations, where variables can be either linear or integer. Commercial MIP solvers (currently CPLEX and Gurobi) are substantially faster than free MIP solvers, and have free academic licenses. MIP solvers can be used directly when the problem is not too large and when the loss function is discrete or linear (e.g., classification error is discrete,

as it takes values either 0 or 1). These solvers are flexible and can handle a huge variety of user-defined constraints easily. However, in the case where the loss function is nonlinear, like the classical logistic loss $\sum_i \log(1 + \exp(-y_i f(x_i)))$, MIP solvers cannot be used directly. For this problem using the logistic loss, Ustun and Rudin (2019) create a method called RiskSLIM that uses sophisticated optimization tools: cutting planes within a branch-and-bound framework, using "callback" functions to a MIP solver. A major benefit of scoring systems is that they can be used as decision aids in very high stakes settings; RiskSLIM has been used to create a model (the 2HELPS2B score) that is used in intensive care units of hospitals to make treatment decisions about critically ill patients (Struck et al., 2017).

While exact optimization approaches provide optimal solutions, they struggle with larger problems. For instance, to handle nonlinearities in continuous covariates, these variables are often discretized to form dummy variables by splitting on all possible values of the covariate (similar to the way continuous variables are handled for logical model construction as discussed above, e.g., create dummy variables for age$<$30, age$<$31, age$<$32, etc.). Obviously doing this can turn a small number of continuous variables into a large number of categorical variables. One way to reduce the size of the problem is to use only a subset of thresholds (e.g., age$<$30, age$<$35, age$<$40, etc.), but it is possible to lose accuracy if not enough thresholds are included. Approximation methods can be valuable in such cases.

Approximate methods. Several works solve approximate versions of (3), including works of Sokolovska et al. (2018, 2017); Billiet et al. (2018, 2017); Carrizosa et al. (2016). These works generally use a piecewise linear or piecewise constant loss function, and sometimes use the ℓ_1 norm for regularization as a proxy for the number of terms. This allows for the possibility of solving linear programs using a mathematical programming solver, which is generally computationally efficient since linear programs can be solved much faster than mixed-integer programs. The main problems with approximation approaches is that it is not clear how well the solution of the approximate optimization problem is to the solution of the desired optimization problem, particularly when user-defined constraints or preferences are required. Some of these constraints may be able to be placed into the mathematical program, but it is still not clear whether the solution of the optimization problem one solves would actually be close to the solution of the optimization problem we actually care about.

It is possible to use sampling to try to find useful scoring systems, which can be useful for Bayesian interpretations, though the space of scoring systems can be quite large and hard to sample (Rudin and Ertekin, 2018).

Sophisticated rounding methods. Keeping in mind the disadvantages to rounding discussed above, there are some compelling advantages to sophisticated rounding approaches, namely that they are easy to program and use in practice. Rounding techniques cannot easily accommodate constraints, but they can be used for problems without constraints, problems where the constraints are weak enough that rounding would not cause us to violate them, or they can be used within the middle of algorithms like RiskSLIM (Ustun and Rudin, 2019) to help find optimal solutions faster. There are several variations of sophisticated rounding. Chevaleyre et al. (2013) propose randomized rounding, where non-integers are rounded up or down randomly. They also propose a greedy method where the sum of coefficients is fixed and coefficients are rounded one at a time. Sokolovska et al. (2018) propose an algorithm that finds a local minimum by improving the solution at each iteration until no further improvements are possible. Ustun and Rudin (2019) propose a combination of

rounding and "polishing." Their rounding method is called Sequential Rounding. At each iteration, Sequential Rounding chooses a coefficient to round and whether to round it up or down. It makes this choice by evaluating each possible coefficient rounded both up and down, and chooses the option with the best objective. After Sequential Rounding produces an integer coefficient vector, a second algorithm, called Discrete Coordinate Descent (DCD), is used to "polish" the rounded solution. At each iteration, DCD chooses a coefficient, and optimizes its value over the set of integers to obtain a feasible integer solution with a better objective. All of these algorithms are easy to program and might be easier to deal with than troubleshooting a MIP or LP solver.

Computer-aided exploration techniques. These are design interfaces where domain experts can modify the model itself rather than relying directly on optimization techniques to encode constraints. Billiet et al. (2018, 2017) created a toolbox that allows users to make manual adjustments to the model, which could potentially help users design interpretable models according to certain types of preferences. Xie et al. (2020) also suggest an expert-in-the-loop approach, where heuristics such as the Gini splitting criteria can be used to help discretize continuous variables. Again, with these approaches, the domain expert must know how they want the model to depend on its variables, rather than considering overall performance optimization.

2.1 **Improve the scalability of optimal sparse scoring systems:** As discussed, for scoring systems, the only practical approaches that produce optimal scoring systems require a MIP solver, and these approaches may not be able to scale to large problems, or optimally handle continuous variables. Current state-of-the-art methods for optimal scoring systems (like RiskSLIM) can deal with a dataset with about thousands of samples and tens of variables within an hour. However, an hour is quite long if one wants to adjust constraints and rerun it several times. How to scale up to large datasets or to reduce solving time remains a challenge, particularly when including complicated sets of constraints.

2.2 **Ease of constraint elicitation and handling:** Since domain experts often do not know the full set of constraints they might want to use in advance, and since they also might want to adjust the model manually (Billiet et al., 2017), a more holistic approach to scoring systems design might be useful. There are many ways to cast the problem of scoring system design with feedback from domain experts. For instance, if we had better ways of representing and exploring the Rashomon set (see Challenge 9), domain experts might be able to search within it effectively, without fear of leaving that set and producing a suboptimal model. If we knew domain experts' views about the importance of features, we should be able to incorporate that through regularization (Wang et al., 2018). Better interfaces might elicit better constraints from domain experts and incorporate such constraints into the models. Faster optimization methods for scoring systems would allow users faster turnaround for creating these models interactively.

Example: A physician wants to create a scoring system for predicting seizures in critically ill patients (similarly to Struck et al., 2017). The physician has upwards of 70 clinical variables, some of which are continuous (e.g., patient age). The physician creates dummy variables for age and other continuous features, combines them with the variables that are already binarized, runs ℓ_1-regularized logistic regression and rounds the coefficients. However, the model does not look reasonable, as it uses only one feature, and isn't very accurate. The physician thinks that the model

should instead depend heavily on age, and have a false positive rate below 20% when the true positive rate is above 70% (see Challenge 2.2). The physician downloads a piece of software for developing scoring systems. The software reveals that a boosted decision tree model is much more accurate, and that there is a set of scoring systems with approximately the same accuracy as the boosted tree. Some of these models do not use age and violate the physician's other constraint, so these constraints are then added to the system, which restricts its search. The physician uses a built-in tool to look at models provided by the system and manually adjusts the dependence of the model on age and other important variables, in a way that still maintains predictive ability. Ideally, this entire process takes a few hours from start to finish (see Challenge 2.1). Finally, the physician takes the resulting model into the clinic to perform a validation study on data the model has not been trained with, and the model is adopted into regular use as a decision aid for physicians.

3 Generalized Additive Models

Generalized additive models (GAMs) were introduced to present a flexible extension of generalized linear models (GLMs) (Nelder and Wedderburn, 1972), allowing for arbitrary functions for modeling the influence of each feature on a response (Hastie and Tibshirani 1990 see also Wood 2017). The set of GAMs includes the set of additive models, which, in turn, includes the set of linear models, which includes scoring systems (and risk scores). Figure 4 shows these relationships.

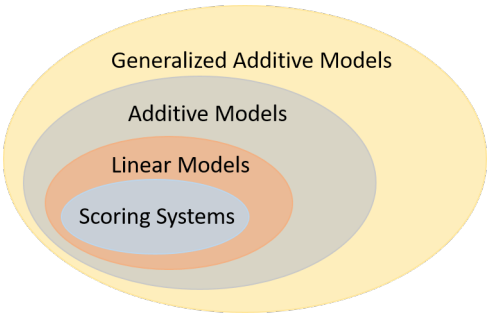


Figure 4: Hierarchical relationships between GAMs, additive models, linear models, and scoring systems.

The standard form of a GAM is

$$g(E[y]) = \beta_0 + f_1(x_{\cdot 1}) + ... + f_p(x_{\cdot p}),$$

where $x_{\cdot j}$ indicates the jth feature, $g(\cdot)$ is a link function and the f_i's are univariate component functions that are possibly nonlinear; common choices are step functions and splines. If the link function $g(\cdot)$ is the identity, the expression describes an additive model such as a regression model; if the link function is the logistic function, then the expression describes a generalized additive model that could be used for classification. The standard form of GAMs is interpretable because the model is constrained to be a linear combination of univariate component functions. We can plot each component function with $f_j(x_{\cdot j})$ as a function of $x_{\cdot j}$ to see the contribution of a single feature to the prediction. The left part of Figure 5 shows all component functions of a GAM model (with no interactions) that predicts whether a patient has diabetes. The right enlarged figure visualizes

the relationship between plasma glucose concentration after 2 hours into an oral glucose tolerance test and the risk of having diabetes.

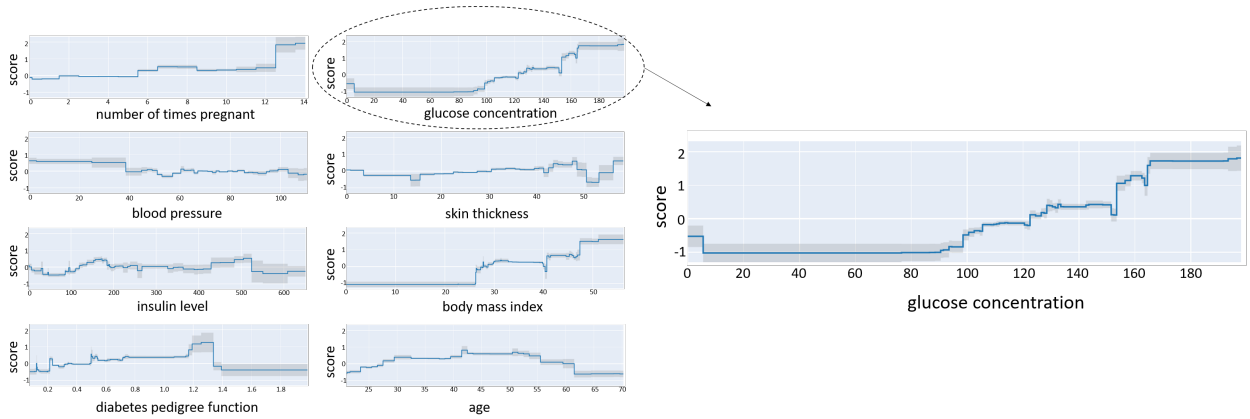


Figure 5: Left: All component functions of a GAM model trained using the `interpret` package (Nori et al., 2019) on a diabetes dataset (Dua and Graff, 2017); Right: zoom-in of component function for glucose concentration.

If the features are all binary (or categorical), the GAM becomes a linear model and the visualizations are just step functions. The visualizations become more interesting for continuous variables, like the ones shown in Figure 5. If a GAM has bivariate component functions (that is, if we choose an f_j to depend on two variables, which permits an interaction between these two variables), a heatmap can be used to visualize the component function on the two dimensional plane and understand the pairwise interactions (Lou et al., 2013). As a comparison point with decision trees, GAMs typically do not handle more than a few interaction terms, and all of these would be quadratic (i.e., involve 2 variables); this contrasts with decision trees, which handle complex interactions of categorical variables. GAMs, like other linear models, do not handle multiclass problems in a natural way. GAMs have been particularly successful for dealing with large datasets of medical records that have many continuous variables because they can elucidate complex relationships between, for instance, age, disease and mortality. (Of course, dealing with large raw medical datasets, we would typically encounter serious issues with missing data, or bias in the labels or variables, which would be challenging for any method, including GAMs.)

A component function f_j can take different forms. For example, it can be a weighted sum of indicator functions, that is:

$$f_j(x_{\cdot j}) = \sum_{\text{thresholds} j'} c_{j,j'} \mathbb{1}[x_{\cdot j} > \theta_{j'}]. \tag{4}$$

If the weights on the indicator functions are integers, and only a small set of weights are nonzero, then the GAM becomes a scoring system. If the indicators are all forced to aim in one direction (e.g., $\mathbb{1}[x_{\cdot j} > \theta_{j'}]$ for all j', with no indicators in the other direction, $\mathbb{1}[x_{\cdot j} < \theta_{j'}]$) and the coefficients $c_{j,j'}$ are all constrained to be nonnegative, then the function will be monotonic. In the case that splines are used as component functions, the GAM can be a weighted sum of the splines' basis functions, i.e. $f_j(x_{\cdot j}) = \sum_{k=1}^{K_j} \beta_{jk} b_{jk}(x_{\cdot j})$.

There are many different ways to fit GAMs. The traditional way is to use backfitting, where we iteratively train a component function to best fit the residuals from the other (already-chosen)

components (Hastie and Tibshirani, 1990). If the model is fitted using boosting methods (Freund and Schapire, 1997; Friedman, 2001, 2002), we learn a tree on each single feature in each iteration and then aggregate them together (Lou et al., 2012). Among different estimations of component functions and fitting procedures, Binder and Tutz (2008) found that boosting performed particularly well in high-dimensional settings, and Lou et al. (2012) found that using a shallow bagged ensemble of trees on a single feature in each step of stochastic gradient boosting generally achieved better performance.

We remark that GAMs have the advantage that they are very powerful, particularly if they are trained as boosted stumps or trees, which are reliable out-of-the-box machine learning techniques. The AdaBoost algorithm also has the advantage that it maximizes convex proxies for both classification error and area under the ROC curve (AUC) simultaneously (Rudin and Schapire, 2009; Şeyda Ertekin and Rudin, 2011). This connection explains why boosted models tend to have both high AUC and accuracy. However, boosted models are not naturally sparse, and issues with bias arise under ℓ_1 regularization, as discussed in the scoring systems section.

We present two interesting challenges involving GAMs:

3.1 **How to control the simplicity and interpretability for GAMs**: The simplicity of GAMs arises in at least two ways: sparsity in the number of component functions and smoothness of the component functions. Imposing monotonicity of the component functions also helps with interpretability when we have prior knowledge, e.g., that risk increases with age. In the case when component functions are estimated by splines, many works apply convex regularizers (e.g., ℓ_1) to control both smoothness and sparsity (Lin et al., 2006; Ravikumar et al., 2009; Meier et al., 2009; Yin and Yu, 2017; Petersen et al., 2016; Lou et al., 2016; Sadhanala et al., 2019; Haris et al., 2019). For example, they add "roughness" penalties and lasso type penalties on the f_j's in the objective function to control both the smoothness of component functions and sparsity of the model. Similarly, if the f_j's are sums of indicators, as in (4), we could regularize to reduce the $c_{j,j'}$ coefficients to induce smoothness. These penalties are usually convex, therefore, when combined with convex loss functions, convex optimization algorithms minimize their (regularized) objectives. There could be some disadvantages to this setup: (1) as we know, ℓ_1 regularization imposes a strong bias on very sparse solutions. (2) Lou et al. (2012) find that imposing smoothness may come at the expense of accuracy, (3) imposing smoothness may miss important naturally-occurring patterns like a jump in a component function; in fact, they found such a jump in mortality as a function of age that seems to occur around retirement age. In that case, it might be more interpretable to include a smooth increasing function of age plus an indicator function around retirement age. At the moment, these types of choices are hand-designed, rather than automated.

As mentioned earlier, boosting can be used to train GAMs to produce accurate models. However, sparsity and smoothness are hard to control with AdaBoost since it adds a new term to the model at each iteration.

3.2 **How to use GAMs to troubleshoot complex datasets?** GAMs are often used on raw medical records or other complex data types, and these datasets are likely to benefit from troubleshooting. Using a GAM, we might find counterintuitive patterns; e.g., as shown in Caruana et al. (2015), asthma patients fared better than non-asthma patients in a health outcomes study. Caruana et al. (2015) provides a possible reason for this finding, which is that asthma patients are

at higher natural risk, and are thus given better care, leading to lower observed risk. Medical records are notorious for missing important information or providing biased information such as billing codes. Could a GAM help us to identify important missing confounders, such as retirement effects or special treatment for asthma patients? Could GAMs help us reconcile medical records from multiple data storage environments? These data quality issues can be really important.

Example: Suppose a medical researcher has a stack of raw medical records and would like to predict the mortality risk for pneumonia patients. The data are challenging, including missing measurements (structural missingness as well as data missing not at random, and unobserved variables), insurance codes that do not convey exactly what happened to the patient, nor what their state was. However, the researcher decides that there is enough signal in the data that it could be useful in prediction, given a powerful machine learning method, such as a GAM trained with boosted trees. There are also several important continuous variables, such as age, that could be visualized. A GAM with a small number of component functions might be appropriate since the doctor can visualize each component function. If there are too many component functions (GAM without sparsity control), analyzing contributions from all of them could be overwhelming (see Challenge 3.1). If the researcher could control the sparsity, smoothness, and monotonicity of the component functions, she might be able to design a model that not only predicts well, but also reveals interesting relationships between observed variables and outcomes. This model could also help us to determine whether important variables were missing, recorded inconsistently or incorrectly, and could help identify key risk factors (see Challenge 3.2).

From there, the researcher might want to develop even simpler models, such as decision trees or scoring systems, for use in the clinic (see Challenges 1 and 2).

4 Modern case-based reasoning

Case-based reasoning is a paradigm that involves solving a new problem using known solutions to similar past problems (Aamodt and Plaza, 1994). It is a problem-solving strategy that we humans use naturally in our decision-making processes (Newell et al., 1972). For example, when ornithologists classify a bird, they will look for specific features or patterns on the bird and compare them with those from known bird species to decide which species the bird belongs to. The interesting question is: can a machine learning algorithm emulate the case-based reasoning process that we humans are accustomed to? A model that performs case-based reasoning is appealing, because by emulating how humans reason, the model can explain its decision-making process in an interpretable way.

The potential uses for case-based reasoning are incredibly broad: whereas the earlier challenges apply only to tabular data, case-based reasoning applies to both tabular and raw data, including computer vision. For computer vision and other raw data problems, we distinguish between the feature extraction and prediction steps. While a human may not be able to understand the full mapping from the original image to the feature (or concept) space, the human may be able to visually verify that a particular interpretable concept/feature has been extracted from an image. After this step, the features/concepts are combined to form a prediction, which is usually done via a sparse linear combination, or with another calculation that a human can understand.

Case-based reasoning has long been a subject of interest in the artificial intelligence (AI) community (Riesbeck and Schank, 1989; Kolodner, 1988; Hammond, 1989). There are, in general, two types of case-based reasoning techniques: (i) nearest neighbor-based techniques, and (ii) prototype-based techniques, illustrated in Figure 6. There are many variations of each of these two types.

Figure 6: Case-based reasoning types. *Left:* Nearest neighbors (just some arrows are shown for 3-nearest neighbors). *Right:* Prototype-based reasoning, shown with two prototypes.

Nearest neighbor-based techniques. These techniques make a decision for a previously unseen test instance, by finding training instances that most closely resemble the particular test instance (i.e., the training instances that have the smallest "distance" or the largest "similarity" measures to the test instance). A classic example of nearest neighbor-based techniques is k-nearest neighbors (kNN) (Fix and Hodges, 1951; Cover and Hart, 1967). Traditionally, a kNN classifier is non-parametric and requires no training at all – given a previously unseen test instance, a kNN classifier finds the k training instances that have the smallest ℓ^2 distances to the test instance, and the class label of the test instance is predicted to be the majority label of those k training instances. Many variants of the original kNN classification scheme have been developed in the 1970s and 80s (Dudani, 1976; Fukunaga and Hostetler, 1973; Keller et al., 1985; Bezdek et al., 1986). Some later papers on nearest neighbor-based techniques focused on the problem of "adaptive kNN," where the goal is to learn a suitable distance metric to quantify the dissimilarity between any pair of input instances (instead of using a pre-determined distance metric such as the Euclidean ℓ^2 distance measure), to improve the performance of nearest neighbor-based techniques. For example, Weinberger and Saul (2009) proposed a method to learn a parametrized distance metric (such as the matrix in a Mahalanobis distance measure) for kNN. Their method involves minimizing a loss function on the training data, such that for every training instance, the distance between the training instance and its "target" neighbors (of the same class) will be minimized while the distance between the training instance and its "imposter" neighbors (from other classes) will be maximized (until those neighbors are at least 1 distance unit away from the training instance). More recently, there are works that began to focus on "performing kNN in a learned latent space," where the latent space is often learned using a deep neural network. For example, Salakhutdinov and Hinton (2007) proposed to learn a nonlinear transformation, using a deep neural network that transforms the input space into a feature space where a kNN classifier will perform well (i.e., deep kNN). Papernot and McDaniel (2018) proposed an algorithm for deep kNN classification, which uses the k-nearest neighbors of a test instance from every hidden layer of a trained neural network. Card et al. (2019) introduced a deep weighted averaging classifier, which classifies an input based on its latent-space distances to other training examples.

We remark that the notion of "adaptive kNN" is mathematically the same as "performing kNN in a learned latent space." Let us show why that is. In adaptive kNN, we would learn a distance metric $d(\cdot, \cdot)$ such that k-nearest neighbors tends to be an accurate classifier. The distance between points x_1 and x_2 would be $d(x_1, x_2)$. For latent space nearest neighbor classification, we would

learn a mapping $\phi : x \to \phi(x)$ from our original space to the latent space, and then compute ℓ_2 distances in the latent space. That is, $\|\phi(x_1), \phi(x_2)\|_2$. Equating these two perspectives, we have $d(x_1, x_2) = \|\phi(x_1), \phi(x_2)\|_2$. That is, the latent space mapping acts as a transformation of the original metric so that ℓ_2 distance works well for kNN.

As an aside, the problem of *matching in observational causal inference* also can use case-based reasoning. Here treatment units are matched to similar control units to estimate treatment effects. We refer readers to recent work on this topic for more details (Wang et al., 2021a; Parikh et al., 2019).

Prototype-based techniques. Despite the popularity of nearest-neighbor techniques, those techniques often require a substantial amount of distance computations (e.g., to find out the nearest neighbors of a test input), which can be slow in practice. Also, it is possible that the nearest neighbors may not be particularly good representatives of a class, so that reasoning about nearest neighbors may not be interpretable. Prototype-based techniques are an alternative to nearest-neighbor techniques that have neither of these disadvantages. Prototype-based techniques learn, from the training data, a set of *prototypical cases* for comparison. Given a previously unseen test instance, they make a decision by finding prototypical cases (instead of training instances from the entire training set) that most closely resemble the particular test instance. One of the earliest prototype learning techniques is learning vector quantization (LVQ) (Kohonen, 1995). In LVQ, each class is represented by one or more prototypes, and points are assigned to the nearest prototype. During training, if the training example's class agrees with the nearest prototype's class, then the prototype is moved closer to the training example; otherwise the prototype is moved further away from the training example. In more recent works, prototype learning is also achieved by solving a discrete optimization program, which selects the "best" prototypes from a set of training instances according to some training objective. For example, Bien and Tibshirani (2011) formulated the prototype learning problem as a set-cover integer program (an NP-complete problem), which can be solved using standard approximation algorithms such as relaxation-and-rounding and greedy algorithms. Kim et al. (2016) formulated the prototype learning problem as an optimization program that minimizes the squared maximum mean discrepancy, which is a submodular optimization problem and can be solved approximately using a greedy algorithm.

Part-based prototypes. One issue that arises with both nearest neighbor and prototype techniques is the comparison of a *whole* observation to another *whole* observation. This makes little sense, for instance, with images, where some aspects resemble a known past image, but other aspects resemble a different image. For example, we consider architecture of buildings: while some architectural elements of a building may resemble one style, other elements resemble another. Another example is recipes. A recipe for a cheesecake with strawberry topping may call for part of a strawberry pancake recipe according to the typical preparation for a strawberry sauce, while the cheesecake part of the recipe could follow a traditional plain cheesecake recipe. In that case, it makes more sense to compare the strawberry cheesecake recipe to both the pancake recipe and the plain cheesecake recipe. Thus, some newer case-based reasoning methods have been comparing *parts* of observations to *parts* of other observations, by creating comparisons on subsets of features. This allows case-based reasoning techniques both more flexibility and more interpretability.

Kim et al. (2014) formulated a prototype-parts learning problem for structured (tabular) data using a Bayesian generative framework. They considered the example (discussed above) of recipes in their experiments. Wu and Tabak (2017) used a convex combination of training instances to represent a prototype, where the prototype does not necessarily need to be a member of the training

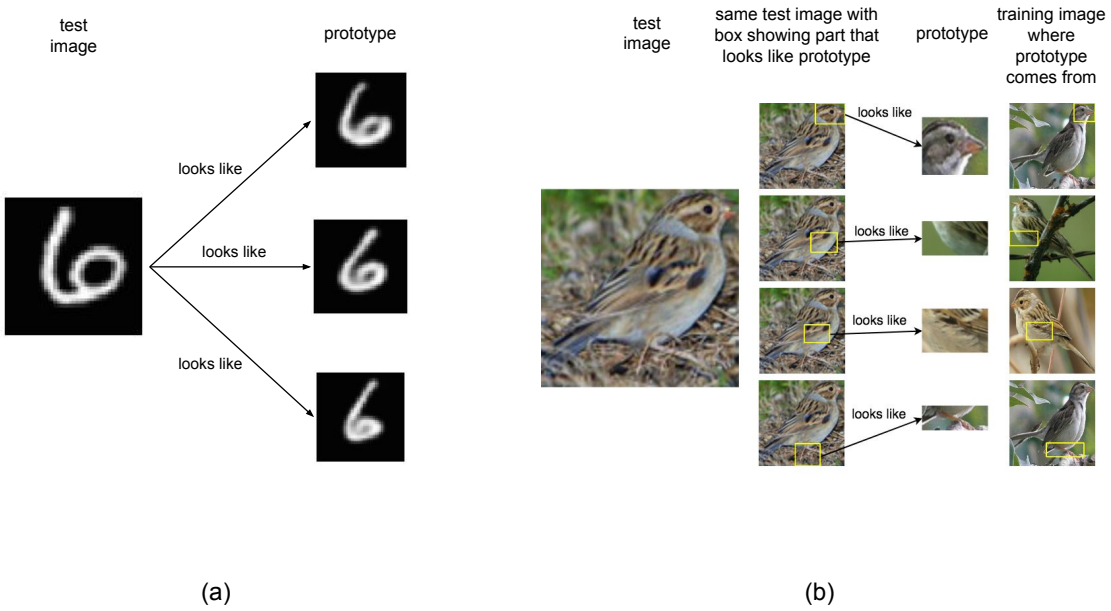


Figure 7: (a) Prototype-based classification of the network in Li et al. (2018). The network compares a previously unseen image of "6" with 15 prototypes of handwritten digits, learned from the training set, and classifies the image as a 6 because it looks like the three prototypes of handwritten 6's, which have been visualized by passing them through a decoder from latent space into image space. (b) Part-based prototype classification of a ProtoPNet in Chen et al. (2019). The ProtoPNet compares a previously unseen image of a bird with prototypical parts of a clay colored sparrow, which are learned from the training set. It classifies the image as a clay colored sparrow because (the network thinks that) its head looks like a prototypical head from a clay-colored sparrow, its wing bars look like prototypical wing bars from a clay-colored sparrow, and so on. Here, the prototypes do not need to be passed through a decoder, they are images from the training set.

set. Using a convex combination of training examples as a prototype would be suitable for some data types (e.g., tabular data, where a convex combination of real training examples might resemble a realistic observation), but for images, averaging the latent positions of units in latent space may not correspond to a realistic-looking image, which means the prototype may not look like a real image, which could be a disadvantage to this type of approach.

Recently, there are works that integrate deep learning with prototype- and prototype-parts-based classification. This idea was first explored by Li et al. (2018) for image classification. In this work, Li et al. created a neural network architecture that contains an autoencoder and a special prototype layer, where each unit of that layer (i.e., a prototype) stores a weight vector that resembles some encoded training input. Given an input instance, the network compares the encoded input instance with the learned prototypes (stored in the prototype layer), which can be visualized using the decoder. The prediction of the network is based on the ℓ^2 distances between the encoded input instance and the learned prototypes. The authors applied the network to handwritten digit recognition (MNIST, LeCun et al., 2010), and the network was able to learn prototypical cases of

how humans write digits by hand. Given a test image of a handwritten digit, the network was also able to find prototypical cases that are similar to the test digit (Figure 7(a)).

More recently, Chen et al. (2019) extended the work of Li et al. (2018) to create a *prototypical part network* (ProtoPNet) whose prototype layer stores *prototypical parts* of encoded training images. The prototypical parts are patches of convolutional-neural-network-encoded training images, and represent typical features observed for various image classes. Given an input instance, the network compares an encoded input image with each of the learned prototypical parts, and generates a *prototype activation map* that indicates both the location and the degree of the image patch most similar to that prototypical part. The authors applied the network to the benchmark Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset (Wah et al., 2011) for bird recognition, and the ProtoPNet was able to learn prototypical parts of 200 classes of birds, and use these prototypes to classify birds with an accuracy comparable to non-interpretable black-box models. Given a test image of a bird, the ProtoPNet was able to find prototypical parts that are similar to various parts of the test image, and was able to provide an explanation for its prediction, such as "this bird is a clay-colored sparrow, because its head looks like that prototypical head from a clay-colored sparrow, and its wing bars look like those prototypical wing bars from a clay-colored sparrow" (Figure 7(b)). In their work, Chen et al. also removed the decoder, and instead introduced *prototype projection*, which pushes every prototypical part to the nearest encoded training patch of the same class for visualization. This improved the visualization quality of the learned prototypes (in comparison to the approach of Li et al. 2018 which used a decoder).

The works of Li et al. (2018) and Chen et al. (2019) have been extended in the domain of deep case-based reasoning and deep prototype learning. In the image recognition domain, Nauta et al. (2020a) proposed a method for explaining what visual characteristics a prototype (in a trained ProtoPNet) is looking for; Nauta et al. (2020b) proposed a method for learning neural prototype trees based on a prototype layer; Rymarczyk et al. (2020) proposed data-dependent merge-pruning of the prototypes in a ProtoPNet, to allow prototypes that activate on similarly looking parts from various classes to be pruned and shared among those classes. In the sequence modeling domain (such as natural language processing), Ming et al. (2019) and Hong et al. (2020) took the concepts in Li et al. (2018) and Chen et al. (2019), and integrated prototype learning into recurrent neural networks for modeling sequential data. Barnett et al. (2021) extended the ideas of Chen et al. (2019) and developed an application to interpretable computer-aided digital mammography.

Despite the recent progress, many challenges still exist in the domain of case-based reasoning, including:

4.1 **How to extend the existing case-based reasoning approach to handling more complex data, such as video?** Currently, case-based reasoning has been used for structured (tabular) data, static images, and simple sequences such as text. It remains a challenge to extend the existing case-based reasoning approach to handling more complex data, such as video data, which are sequences of static images. While Trinh et al. (2021) recently extended the ideas in Chen et al. (2019) and developed a dynamic prototype network (DPNet) which learns prototypical video patches from deep-faked and real videos, how to efficiently compare various videos and find similar videos (for either nearest-neighbor or prototype-based classification) remains an open challenge for case-based video classification tasks. Performing case-based reasoning on video data is technically challenging because of the high dimensionality of the input data. On the other hand, a video is an ordered combination of frames and we can take ad-

vantage of the sequential nature of the data. For example, current algorithms can be refined or improved with more information that comes from neighboring video frames; could prototypes be designed from neighboring frames or parts of the frames?

4.2 **How to integrate prior knowledge or human supervision into prototype learning?** Current approaches to prototype learning do not take into account prior knowledge or expert opinions. At times, it may be advantageous to develop prototype-learning algorithms that collaborate with human experts in choosing prototypical cases or prototypical features. For example, in healthcare, it would be beneficial if a prototype-based classifier learns, under the supervision of human doctors, prototypical signs of cancerous growth. For example, the doctors might prune prototypes, design them, or specify a region of interest where the prototypes should focus. Such human-machine collaboration would improve the classifier's accuracy and interpretability, and would potentially reduce the amount of data needed to train the model. However, human-machine collaboration is rarely explored in the context of prototype learning, or case-based reasoning in general.

4.3 **How to troubleshoot a trained prototype-based model to improve the quality of prototypes?** Prototype-based models make decisions based on similarities with learned prototypes. However, sometimes, the prototypes may not be learned well enough, in the sense that they may not capture the most representative features of a class. This is especially problematic for part-based prototype models, because these models reason by comparing subsets of features they deem important. In the case of a (part-based) prototype model with "invalid" prototypes that capture non-representative or undesirable features (e.g., a prototype of text in medical images, which represents information that improves training accuracy but not test accuracy), one way to "fix" the model is to get rid of the invalid prototypes, but this may lead to an imbalance in the number of prototypes among different classes and a bias for some classes that are over-represented with abundant prototypes. An alternative solution is to replace each undesirable prototype with a different prototype that does not involve the undesirable features. The challenge here lies in how we can replace undesirable prototypes systematically without harming the model performance, and at the same time improve the given model incrementally without retraining the model from scratch.

Example: Suppose that a doctor wants to evaluate the risk of breast cancer among patients (see Barnett et al., 2021). The dataset used to predict malignancy of breast cancer usually contains a set of mammograms, and a set of patient features (e.g., age). Given a particular patient, how do we characterize prototypical signs of cancerous growth from a sequence of mammograms taken at various times (this is similar to a video, see Challenge 4.1)? How can a doctor supervise prototype learning by telling a prototype-based model what (image/patient) features are typical of breast cancer (Challenge 4.2)? If a trained model contains a prototype of "text" in the mammograms (such text could be doctor's notes or patient IDs left on some training images), how can we replace the unwanted prototype with another prototype that is more medically relevant, without retraining the model from scratch (Challenge 4.3)?

5 Complete supervised disentanglement of neural networks

Deep neural networks (DNNs) have achieved state-of-the-art predictive performance for many important tasks (LeCun et al., 2015). DNNs are the quintessential "black box" because the computations within its hidden layers are typically inscrutable. As a result, there have been many works that have attempted to "disentangle" DNNs in various ways so that information flow through the network is easier to understand. "Disentanglement" here refers to the way information travels through the network: we would perhaps prefer that all information about a specific concept (say "lamps") traverse through one part of the network while information about another concept (e.g., "airplane") traverse through a separate part. Therefore, disentanglement is an interpretability constraint on the neurons. For example, suppose we hope neuron "neur" in layer l is aligned with concept c, in which case, the disentanglement constraint is

$$\text{DisentanglementConstraint}(c, \text{neur}, l) : \text{Signal}(c, \text{neur}) = \text{Signal}(c, l), \tag{5}$$

where $\text{Signal}(c, x)$ means the signal of concept c that passes through x, which measures similarity between its two arguments. This constraint means *all* signal about concept c in layer l will *only* pass through neuron neur. In other words, using these constraints, we could constrain our network to have the kind of "Grandmother node" that scientists have been searching for in both real and artificial convolutional neural networks (Gross, 2002).

In this challenge, we consider the possibility of fully disentangling a DNN so that each neuron in a piece of the network represents a human-interpretable concept.

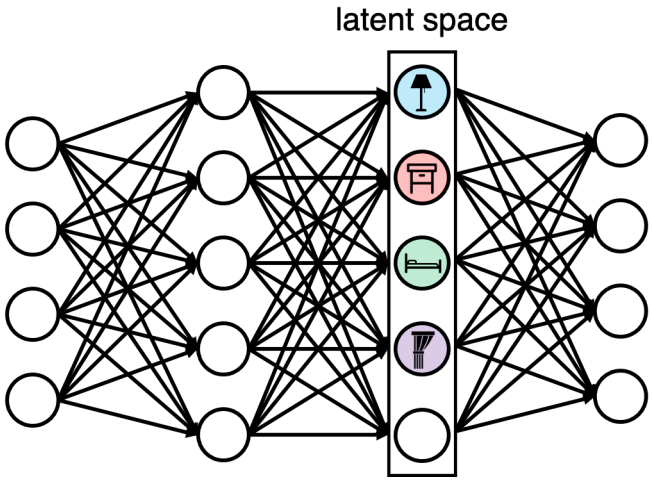


Figure 8: Disentangled latent space. The axes (neurons) of the latent space are aligned with supervised concepts, e.g. "lamp," "bed," "nightstand," "curtain." All information about the concept up to that point in the network travels through that concept's corresponding neuron.

The vector space whose axes are formed by activation values on the hidden layer's neurons is known as the latent space of a DNN. Figure 8 shows an example of what an ideal interpretable latent space might look like. The axes of the latent space are aligned with individual visual concepts, such as "lamp," "bed," "nightstand," "curtain." Note that the "visual concepts" are not restricted to objects but can also be things such as weather or materials in a scene. We hope all information

about the concept travels through that concept's corresponding neuron on the way to the final prediction. For example, the "lamp" neuron will be activated if and only if the network thinks that the input image contains information about lamps. This kind of representation makes the reasoning process of the DNN much easier to understand: the image is classified as "bedroom" because it contains information about "bed" and "lamp."

Such a latent space, made of disjoint data generation factors, is a disentangled latent space (Bengio, 2009; Higgins et al., 2018). An easy way to create a disentangled latent space is just to create a classifier for each concept (e.g., create a lamp classifier), but this is not a good strategy: it might be that the network only requires the light of the lamp rather than the actual lamp body, so creating a lamp classifier could actually reduce performance. Instead, we would want to encourage the information that is used about a concept to go along one path through the network.

Disentanglement is not guaranteed in standard neural networks. In fact, information about any concept could be scattered throughout the latent space of a standard DNN. For example, post hoc analyses on neurons of standard convolutional neural networks (Zhou et al., 2018a, 2014) show that concepts that are completely unrelated could be activated on the same axis, as shown in Figure 9. Even if we create a vector in the latent space that is aimed towards a single concept (as is done in Kim et al., 2018; Zhou et al., 2018b), that vector could activate highly on multiple concepts, which means the signal for the two concepts is not disentangled. In that sense, vectors in the latent space are "impure" in that they do not naturally represent single concepts (see Chen et al., 2020, for a detailed discussion).

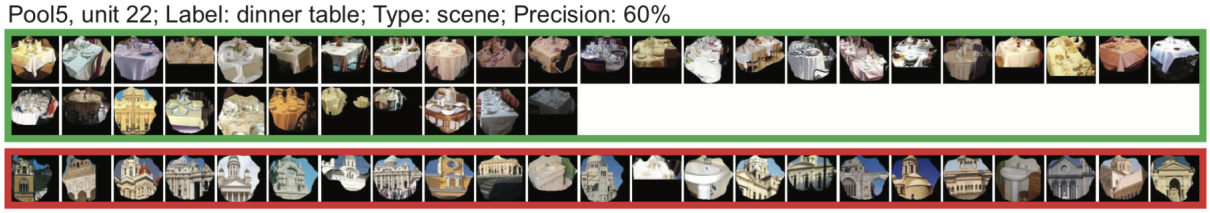


Figure 9: Example of an impure neuron in standard neural network from Zhou et al. (2014). This figure shows images that highly activate this neuron. Both dining tables (green) and Greek-style buildings (red) are highly activated on the neuron, even though these two concepts are unrelated.

This challenge focuses on supervised disentanglement of neural networks, i.e., the researcher specifies which concepts to disentangle in the latent space. (In the next section we will discuss unsupervised disentanglement.) Earlier work in this domain disentangles the latent space for specific applications, such as face recognition, where we might aim to separate identity and pose (Zhu et al., 2014). Recent work in this area aims to disentangle the latent space with respect to a collection of predefined concepts (Chen et al., 2020; Koh et al., 2020; Losch et al., 2019; Adel et al., 2018). For example, Chen et al. (2020) adds constraints to the latent space to decorrelate the concepts and align them with axes in the latent space; this is called "concept whitening." (One could think of this as analogous to a type of principal components analysis for neural networks.) To define the concepts, a separate dataset is labeled with concept information (or the main training dataset could also serve as this separate dataset as long as it has labels for each concept.) These concept datasets are used to align the network's axes along the concepts. This type of method can create disentangled latent spaces where concepts are more "pure" than those of a standard DNN, without hurting accuracy. With the disentangled latent space, one can answer questions about how

the network gradually learns concepts over layers (Chen et al., 2020), or one could interact with the model by intervening on the activations of concept neurons (Koh et al., 2020). Many challenges still exist for supervised disentanglement of DNNs, including:

5.1 **How to make a whole layer of a DNN interpretable?** Current methods can make neurons in a single layer interpretable but all of them have limitations. Koh et al. (2020), Losch et al. (2019) and Adel et al. (2018) try to directly learn a concept classifier in the latent space. However, as discussed by Chen et al. (2020), good discriminative power on concepts does not guarantee concept separation and disentanglement. That is, we could easily have our network classify all the concepts correctly, in addition to the overall classification task, but this would not mean that the information flow concerning each concept goes only though that concept's designated path through the network; the activations of the different concept nodes would likely still be correlated. The method discussed earlier, namely that of Chen et al. (2020), can fully disentangle the latent space with respect to a few pre-defined concepts. But it also has two disadvantages if we want to disentangle all neurons in a layer, namely: (i) this method currently has a group of unconstrained neurons to handle residual information, which means it does not disentangle all neurons in a layer, just some of them, and (ii) it requires loading samples of all target concepts to train the latent space. If the layer to be disentangled contains 500 neurons, we would need to load samples for all 500 concepts, which takes a long time. Perhaps the latter problem could be solved by training with a small random subset of concepts at each iteration; but this has not been tried in current methods at the time of this writing. The first problem might be solved by using an unsupervised concept detector from Challenge 6 and having a human interact with the concepts to determine whether each one is interpretable.

5.2 **How to disentangle all neurons of a DNN simultaneously?** Current methods try to disentangle at most a single layer in the DNN (that is, they attempt only the problem discussed above). This means one could only interpret neurons in that specific layer, while semantic meanings of neurons in other layers remain unknown. Ideally, we want to be able to completely understand and modify the information flowing through all neurons in the network. This is a challenging task for many obvious reasons, the first one being that it is hard practically to define what all these concepts could possibly be. We would need a comprehensive set of human-interpretable concepts, which is hard to locate, create, or even to parameterize. Even if we had this complete set, we would probably not want to manually specify exactly what part of the network would be disentangled with respect to each of these numerous concepts. For instance, if we tried to disentangle the same set of concepts in all layers, it would be immediately problematic because DNNs are naturally hierarchical: high-level concepts (objects, weather, etc.) are learned in deeper layers, and the deeper layers leverage low-level concepts (color, texture, object parts, etc.) learned in lower layers. Clearly, complex concepts like "weather outside" could not be learned well in earlier layers of the network, so higher-level concepts might be reserved for deeper layers. Hence, we also need to know the hierarchy of the concepts to place them in the correct layer. Defining the concept hierarchy manually is almost impossible, since there could be thousands of concepts. But how to automate it is also a challenge.

5.3 **How to choose good concepts to learn for disentanglement?** In supervised disentanglement, the concepts are chosen manually. To gain useful insights from the model, we need good concepts. But what are good concepts in specific application domains? For example, in

medical applications, past works mostly use clinical attributes that already exist in the datasets. However, Chen et al. (2020) found that attributes in the ISIC dataset might be missing the key concept used by the model to classify lesion malignancy. Active learning approaches could be incredibly helpful in interfacing with domain experts to create and refine concepts.

Moreover, it is challenging to learn concepts with continuous values. These concepts might be important in specific applications, e.g., age of the patient and size of tumors in medical applications. Current methods either define a concept by using a set of representative samples or treat the concept as a binary variable, where both are discrete. Therefore, for continuous concepts, a challenge is how to choose good thresholds to transform the continuous concept into one or multiple binary variables.

5.4 **How to make the mapping from the disentangled layer to the output layer interpretable?** The decision process of current disentangled neural networks contains two parts, $x \to c$ mapping the input x to the disentangled representation (concepts) c, and $c \to y$ mapping the disentangled representation c to the output y. (The notations are adopted from Koh et al., 2020). All current methods on neural disentanglement aim at making the c interpretable, i.e., making the neurons in the latent space aligned with human understandable concepts, but how these concepts combined to make the final prediction, i.e., $c \to y$, often remains a black box. This leaves a gap between the interpretability of the latent space and the interpretability of the entire model. Current methods either rely on variable importance methods to explain $c \to y$ posthoc (Chen et al., 2020), or simply make $c \to y$ a linear layer (Koh et al., 2020). However, a linear layer might not be expressive enough to learn $c \to y$. Koh et al. (2020) also shows that a linear function $c \to y$ is less effective than nonlinear counterparts when the user wants to intervene in developing the disentangled representation, e.g., replacing predicted concept values $\hat{c}_j$ with true concept values c_j. Neural networks like neural additive models (Agarwal et al., 2020) and neural decision trees (Yang et al., 2018) could potentially be used to model $c \to y$, since they are both differentiable, nonlinear, and intrinsically interpretable once the input features are interpretable.

Example: Suppose machine learning practitioners and doctors want to build a supervised disentangled DNN on X-ray data to detect and predict arthritis. First, they aim to choose a set of relevant concepts that have been assessed by doctors (Challenge 5.3). They should also choose thresholds to turn continuous concepts (e.g., age) into binary variables to create the concept datasets (Challenge 5.3). Using the concept datasets, they can use supervised disentanglement methods like concept whitening (Chen et al., 2020) to build a disentangled DNN. However, if they choose too many concepts to disentangle in the neural network, loading samples from all of the concept datasets may take a very long time (Challenge 5.1). Moreover, doctors may have chosen different levels of concepts, such as bone spur (high-level) and shape of joint (low-level), and they would like the low-level concepts to be disentangled by neurons in earlier layers, and high-level concepts to be disentangled in deeper layers, since these concepts have a hierarchy according to medical knowledge. However, current methods only allow placing them in the same layer (Challenge 5.2). Finally, all previous steps can only make neurons in the DNN latent space aligned with medical concepts, while the way in which these concepts combine to predict arthritis remains uninterpretable (Challenge 5.4).

6 Unsupervised disentanglement of neural networks

The major motivation of unsupervised disentanglement is the same as Challenge 5, i.e., making information flow through the network easier to understand and interact with. But in the case where we do not know the concepts, or in the case where the concepts are numerous and we do not know how to parameterize them, we cannot use the techniques from Challenge 5. In other words, the concept c in Constraint (5) is no longer a concept we predefine, but it must still be an actual concept in the existing universe of concepts. There are situations where concepts are actually unknown; for example, in materials science, the concepts, such as key geometric patterns in the unit cells of materials, have generally not been previously defined. There are also situations where concepts are generally known but too numerous to handle; a key example is computer vision for natural images. Even though we could put a name to many concepts that exist in natural scenes, labeled datasets for computer vision have a severe labeling bias: we tend only to label entities in images that are useful for a specific task (e.g., object detection), thus ignoring much of the information found in images. If we could effectively perform unsupervised disentanglement, we can rectify problems caused by human bias, and potentially make scientific discoveries in uncharted domains. For example, an unsupervised disentangled neural network can be used to discover key patterns in materials and characterize their relation to the physical properties of the material (e.g., "will the material allow light to pass through it?"). Figure 10 shows such a neural network, with a latent space completely disentangled and aligned with the key patterns discovered without supervision: in the latent space, each neuron corresponds to a key pattern and all information about the pattern flows through the corresponding neuron. Analyzing these patterns' contribution to the prediction of a desired physical property could help material scientists understand what correlates with the physical properties and could provide insight into the design of new materials.

Multiple branches of works are related to unsupervised disentanglement of deep neural networks, and we will describe some of them.

Disentangled representations in deep generative models: Disentanglement of generative models have long been studied (Schmidhuber, 1992; Desjardins et al., 2012). Deep generative models such as generative adversarial networks (GANs, Goodfellow et al., 2014) and variational auto-encoders (VAE, Kingma and Welling, 2013) try to learn a mapping from points in one probability distribution to points in another. The first distribution is over points in the latent space, and these points are chosen i.i.d. according to a zero-centered Gaussian distribution. The second distribution is over the space from which the data are drawn (e.g., random natural images in the space of natural images). The statistical independence between the latent features makes it easy to disentangle the representation (Bengio et al., 2013): a disentangled representation simply guarantees that knowing or changing one latent feature (and its corresponding concept) does not affect the distribution of any other. Results on simple imagery datasets show that these generative models can decompose the data generation process into disjoint factors (for instance, age, pose, identity of a person) and explicitly represent them in the latent space, without any supervision on these factors; this happens based purely on statistical independence of these factors in the data. Recently, the quality of disentanglement in deep generative models has been improved (Chen et al., 2016; Higgins et al., 2017). These methods achieve full disentanglement without supervision by maximizing the mutual information between latent variables and the observations. However, these methods only work for relatively simple imagery data, such as faces or single 3D objects (that is, in the image, there is only one object, not a whole scene). Learning the decomposition of a scene into groups of objects

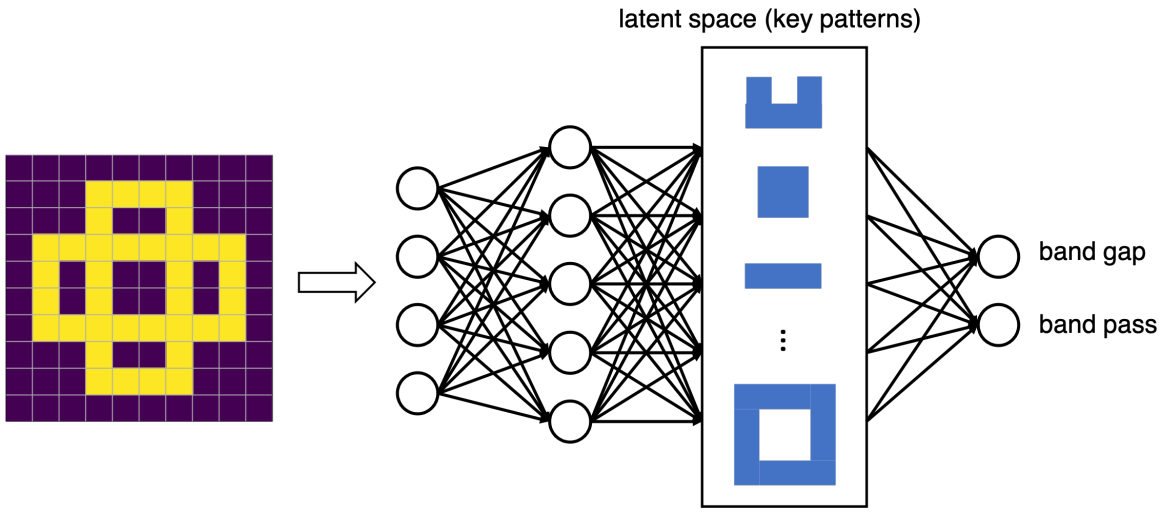


Figure 10: Neural network with unsupervised disentanglement in the latent space. The input (on the left) is a unit cell of metameterial, made of stiff (yellow) and soft (purple) constituent materials. The target of the neural network is predicting whether the material with unit cell on the left supports the formation of forbidden frequency bands of propagation, i.e. existence of "band gaps." Key patterns related to the band gap (illustrated in blue) have been discovered without supervision. Neurons in the latent space are aligned with these patterns. Figure adapted from Chen et al. (2021b).

in the latent space, for example, is not yet achievable by these deep generative models. One reason for the failure of these approaches might be that the occurrence of objects may not be statistically independent; for example, some objects such as "bed" and "lamp" tend to co-occur in the same scene. Also, the same type of object may occur multiple times in the scene, which cannot be easily encoded by a single continuous latent feature.

Neural networks that incorporate compositional inductive bias: Another line of work designs neural networks that directly build compositional structure into a neural architecture. Compositional structure occurs naturally in computer vision data, as objects in the natural world are made of parts. In areas that have been widely studied beyond computer vision, including speech recognition, researchers have already summarized a series of compositional hypotheses, and incorporated them into machine learning frameworks. For example, in computer vision, the "vision as inverse graphics" paradigm (Baumgart, 1974) tries to treat vision tasks as the inverse of the computer graphics rendering process. In other words, it tries to decode images into a combination of features that might control rendering of a scene, such as object position, orientation, texture and lighting. Many studies on unsupervised disentanglement have been focused on creating this type of representation because it is intrinsically disentangled. Early approaches toward this goal includes DC-IGN (Kulkarni et al., 2015) and Spatial Transformers (Jaderberg et al., 2015). Recently, Capsule Networks (Hinton et al., 2011; Sabour et al., 2017) have provided a new way to incorporate compositional assumptions into neural networks. Instead of using neurons as the building blocks, Capsule Networks combine sets of neurons into larger units called "Capsules," and force them to represent information such as pose, color and location of either a particular part or a complete object. This method was later combined with generative models in the Stack Capsule Autoencoder

(SCAE) (Kosiorek et al., 2019). With the help of the Set Transformer (Lee et al., 2019) in combining information between layers, SCAE discovers constituents of the image and organizes them into a smaller set of objects. Slot Attention modules (Locatello et al., 2020) further use an iterative attention mechanism to control information flow between layers, and achieve better results on unsupervised object discovery. Nevertheless, similar to the generative models, these networks perform poorly when aiming to discover concepts on more realistic datasets. For example, SCAE can only discover stroke-like structures that are uninterpretable to humans on the Street View House Numbers (SVHN) dataset (Netzer et al., 2011) (see Figure 11). The reason is that the SCAE can only discover visual structures that appear frequently in the dataset, but in reality the appearance of objects that belong to the same category can vary a lot. There have been proposals (e.g., GLOM Hinton, 2021) on how a neural network with a fixed architecture could potentially parse an image into a part-whole hierarchy. Although the idea seems to be promising, no working system has been developed yet. There is still a lot of room for development for this type of method.

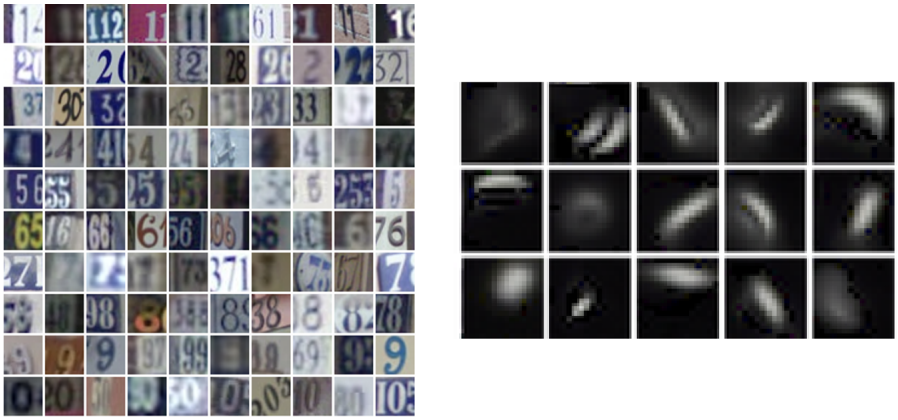


Figure 11: Left: Sample Images collected from the SVHN dataset (Netzer et al., 2011). Right: Stroke-like templates discovered by the SCAE. Although the objects in the dataset are mostly digits, current capsule networks are unable to discover them as concepts without supervision.

Works that perform unsupervised disentanglement implicitly: Many other interpretable neural networks can also learn disjoint concepts in the latent space although the idea of "disentanglement" is not explicitly mentioned in these papers. For example, as mentioned in Section 4, Chen et al. (2019) propose to create a protopype layer, storing prototypical parts of training images, to do case based reasoning. When classifying birds, the prototypical parts are usually object parts such as head and wings of birds. Interestingly, a separation cost is applied to the learned prototypes to encourage diversity of the prototypical parts, which is very similar to the idea of disentanglement. Zhang et al. (2018) propose to maximize mutual information between the output of convolution filters and predefined part templates (feature masks with positive values in a localized region and negative in other places). The mutual information regularization term is essentially the sum of two disentanglement losses, (a) an inter-category entropy loss that encourages each filter to be exclusively activated by images of one category and not activated on other categories; (b) a spatial entropy loss that encourages each filter to be activated only on a local region of the image. Results show the convolutional filters with the mutual information loss tend to be activated only on a specific part of the object. These two methods are capable of learning single objects or parts in the latent space but have not yet been generalized to handle more comprehensive concepts (such

as properties of scenes, including style of an indoor room - e.g., cozy, modern, etc., weather for outdoor scenes, etc.), since these concepts are not localized in the images.

Despite multiple branches of related work targeting concept discovery in different way, many challenges still exist:

6.1 How to quantitatively evaluate unsupervised disentanglement?

Recall that unsupervised disentanglement is desired (but challenging) in two different domains: (a) domains in which we do not know what the concepts are (e.g., materials science); (b) domains in which concepts are known but labeling biases exist.

Let us start with a version of case (b) where some concepts are known, such as objects in natural images. Let us say we are just missing a subset of concept labels in the dataset, which is a simple type of missing data bias. If we build a neural network that disentangles the latent space, we can quantitatively evaluate the quality of disentanglement. For instance, let us say we know a collection of concepts that we would like disentangled (say, a collection of specific objects that appear in natural images). If we use an unsupervised algorithm for disentangling the space, and then evaluate whether it indeed disentangled the known concepts, then we have a useful quantitative evaluation (Higgins et al., 2017; Chen et al., 2018; Kim and Mnih, 2018; Eastwood and Williams, 2018; Do and Tran, 2019).

If we are working in a domain where we do not know the concepts to disentangle (case (a)), we also might not know what regularization (or other type of inductive bias) to add to the algorithm so that it can discover concepts that we would find interpretable. In that case, it becomes difficult to evaluate the results quantitatively. Materials science, as discussed above, is an example of one of these domains, where we cannot find ground-truth labels of key patterns, nor can we send queries to human evaluators (even material scientists do not know the key patterns in many cases). Evaluation metrics from natural images do not work. Evaluating disentanglement in these domains is thus a challenge.

Going back to case (b), in situations where labeling biases exist (i.e., when only some concepts are labeled, and the set of labels are biased), current evaluation metrics of disentanglement that rely only on human annotations can be problematic. For instance, most annotations in current imagery datasets concern only objects in images but ignore useful information such as lighting and style of furniture. An unsupervised neural network may discover important types of concepts that do not exist in the annotations. Medical images may be an important domain here, where some concepts are clearly known to radiologists, but where labels in available datasets are extremely limited.

6.2 How to adapt neural network architectures designed with compositional constraints to other domains?

Incorporating compositional assumptions into network architecture design is a way to create intrinsically interpretable disentangled representations. That said, such assumptions are usually modality- and task-specific, severely limiting the general applicability for such designs. Let us take Capsule Networks (Sabour et al., 2017) and the Slot Attention module (Locatello et al., 2020) in the computer vision field as examples: these modules try to create object-centric and part-centric representations inside the latent space of the network. These ideas are based on the assumption that an image can be understood as a composition of different objects, and

the interference between objects is negligible. Nevertheless, such an assumption cannot necessarily be applied to materials science, in which a material's physical properties could depend jointly, in a complex way, on the patterns of constituent materials within it. How would we redefine the compositional constraints derived in computer vision for natural images to work for other domains such as materials science?

6.3 **How to learn part-whole disentanglement for more complicated patterns in large vision datasets?** A specific area within unsupervised disentanglement in computer vision is to semantically segment the image into different parts, where each part represents an object or a part of an object (Crawford and Pineau, 2019; Sabour et al., 2017; Kosiorek et al., 2019; Locatello et al., 2020). Current networks can achieve convincing results over simple, synthetic datasets (where objects consist of simple geometric shapes or digits). (Such datasets include CLEVR6, Johnson et al. 2017, d-Sprite, Matthey et al. 2017, and Objects Room and Tetrominoes, Kabra et al. 2019.) However, no work has successfully learned part-whole relationships on more realistic datasets such as ImageNet. New techniques may be needed to handle various interactions between different objects in a real-world dataset.

Example: Suppose material scientists want to build a classification model that can predict whether the designs of metameterials support existence of band gaps (same example as Figure 10). Because unit cells of metamaterials are usually represented as a matrix/tensor of what constituent materials are placed at each location, the researchers plan to build a deep neural network, since neural networks excel at extracting useful information from raw inputs. They might encounter several challenges when building the disentangled neural network. First, they need to identify the disentanglement constraints to build into the network architecture. Architectures that work well for imagery data may not work for unit cells of metamaterials, since key patterns in unit cells can be completely different from objects/concepts in imagery data (Challenge 6.2 above). Moreover, evaluation of disentanglement can be challenging as the material patterns have no ground truth labels (Challenge 6.1 above).

7 Dimension Reduction for Data Visualization

Even in data science, a picture is worth a thousand words. Dimension reduction (DR) techniques take, as input, high-dimensional data and project it down to a lower-dimensional space (usually 2D or 3D) so that a human can better comprehend it. Data visualization can provide an intuitive understanding of the underlying structure of the dataset. DR can help us gain insight and build hypotheses. DR can help us design features so that we can build an interpretable supervised model, allowing us to work with high-dimensional data in a way we would not otherwise be able to. With DR, biases or pervasive noise in the data may be illuminated, allowing us to be better data caretakers. However, with the wrong DR method, information about the high-dimensional relationships between points can be lost when projecting onto a 2D or 3D space.

DR methods are unsupervised ML methods that are constrained to be interpretable. Referring back to our generic interpretable ML formulation (1), DR methods produce a function mapping data $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ from p-dimensional space to $\mathbf{y}_1$, $\mathbf{y}_2$, ... $\mathbf{y}_n$ in a low dimensional space (usually 2D). The constraint of mapping to 2 dimensions is an interpretability constraint. DR methods

typically have a loss function that aims to preserve a combination of distance information between points and neighborhood information around each point when projecting to 2D. Each DR algorithm chooses the loss function differently. Each algorithm also chooses its own distance or neighborhood information to preserve.

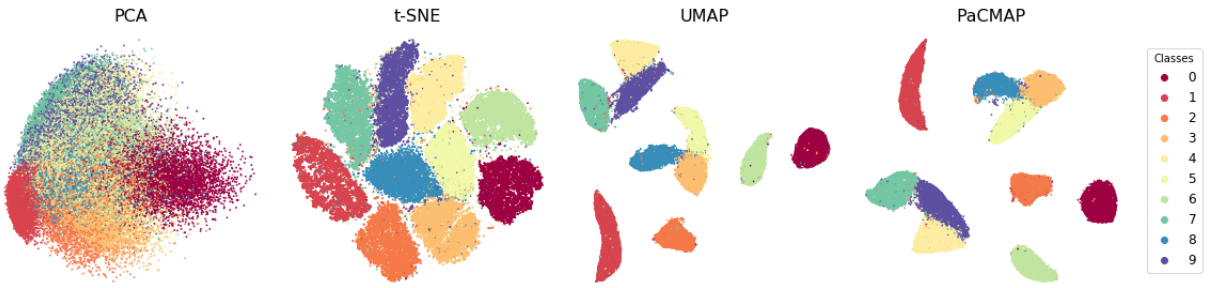


Figure 12: Visualization of the MNIST dataset (LeCun et al., 2010) using different kinds of DR methods: PCA (Pearson, 1901), t-SNE (van der Maaten and Hinton, 2008; Linderman et al., 2019; Poličar et al., 2019), UMAP (McInnes et al., 2018), and PaCMAP (Wang et al., 2020b). The axes are not quantified because these are projections into an abstract 2D space.

Generally speaking, there are two primary types of approaches to DR for visualization, commonly referred to as local and global methods. Global methods aim mainly to preserve distances between any pair of points (rather than neighborhoods), while the local methods emphasize preservation of local neighborhoods (that is, which points are nearest neighbors). As a result, local methods can preserve the local cluster structure better, while failing to preserve the overall layout of clusters in the space, and vice versa. Figure 12 demonstrates the difference between the two kinds of algorithms over the MNIST handwritten figure dataset (LeCun et al., 2010), which is a dataset where local structure tends to be more important than global structure. The only global method here, PCA (Pearson, 1901), fails to separate different digits into clusters, but it gives a sense of which digits are different from each other. t-SNE (van der Maaten and Hinton, 2008), which is a local method, successfully separated all the digits, but could not also keep the scale information that is preserved in the PCA embedding. More recent methods, such as UMAP (McInnes et al., 2018) and PaCMAP (Wang et al., 2020b) also separated the digits while preserving some of the global information.

Early approaches toward this problem, including Principal Component Analysis (PCA) (Pearson, 1901) and Multidimensional Scaling (MDS) (Torgerson, 1952), mostly fall into the global category. They aim to preserve as much information as possible from the high-dimensional space, including the distances or rank information between pairs of points. These methods usually apply matrix decomposition over the data or pairwise distance matrix, and are widely used for data preprocessing. These methods usually fail to preserve local structure, including cluster structure.

To solve these problems, researchers later (early 2000s) developed methods that aimed at local structure, because they had knowledge that high dimensional data lie along low-dimensional manifolds. Here, it was important to preserve the local information along the manifolds. Isomap (Tenenbaum et al., 2000), Local Linear Embedding (LLE) (Roweis and Saul, 2000), Hessian Local Linear Embedding (Donoho and Grimes, 2003), and Laplacian Eigenmaps (Belkin and Niyogi, 2001) all try to preserve exact local Euclidean distances from the original space when creating

low-dimensional embeddings. But distances between points behave differently in high dimensions than in low dimensions, leading to problems preserving the distances. In particular, these methods tended to exhibit what is called the "crowding problem," where samples in the low-dimensional space crowd together, devastating the local neighborhood structure and losing information. t-SNE (van der Maaten and Hinton, 2008) was able to handle this problem by transforming the high-dimensional distances between each pair of points into probabilities of whether the two points should be neighbors in the low-dimensional space. Doing this aims to ensure that local neighborhood structure is preserved. Then, during the projection to low dimensions, t-SNE enforces that the distribution over distances between points in the low-dimensional space is a specific transformation of the distribution over distances between points in the high-dimensional space. Forcing distances in the low-dimensional space to follow a specific distribution avoids the crowding problem, which is when a large proportion of the distances are almost zero.

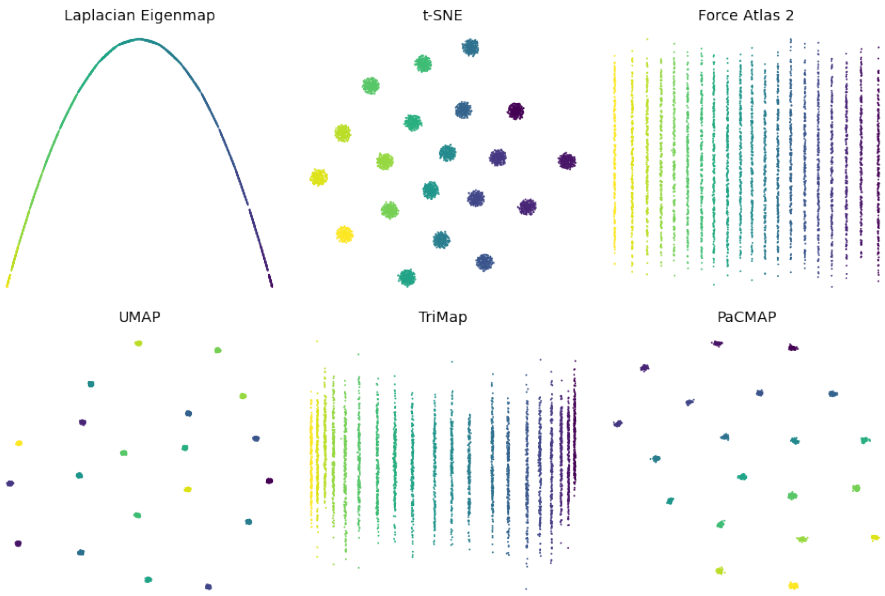


Figure 13: Visualization of 4000 points sampled from 20 isotropic Gaussians using Laplacian Eigenmap (Belkin and Niyogi, 2001; Pedregosa et al., 2011), t-SNE (van der Maaten and Hinton, 2008; Linderman et al., 2019; Poličar et al., 2019), ForceAtlas2 (Jacomy et al., 2014), UMAP (McInnes et al., 2018), PaCMAP (Wang et al., 2020b) and TriMap (Amid and Warmuth, 2019). The 20 Gaussians are equally spaced on an axis in 50-dimensional space, labelled by the gradient colors. The best results are arguably those of t-SNE and PaCMAP in this figure, which preserve clusters compactly and their relative placement (yellow on the left, purple on the right).

Though more successful than previous methods, t-SNE still suffers from slow running times, sensitivity to hyperparameter changes, and lack of preservation of global structure. A series of t-SNE variants aim to improve on these shortcomings. The most famous among them are BH t-SNE (van der Maaten, 2014) and FIt-SNE (Linderman et al., 2019). BH t-SNE constructs a k-nearest neighbor graph over the high dimensional space to record local neighborhood structure, and utilizes the Barnes-Hut force calculation algorithm (which is typically used for multi-body simulation; Barnes and Hut, 1986) to accelerate optimization of the low-dimensional embedding.

These choices reduced the complexity for each step in the optimization from $O(n^2)$, where n is the number of samples in the dataset, to a smaller $O(n \log n)$. FIt-SNE further accelerates the rendering phase with a Fast Fourier Transform, and brings down the complexity to $O(n)$. Besides these variants, multiple new algorithms have been created based on the framework of t-SNE. Prominent examples include LargeVis (Tang et al., 2016), which is widely used in network analysis, and UMAP (McInnes et al., 2018), which is widely used in computational biology. With a better initialization created by unsupervised machine learning algorithms (such as spectral clustering) and better loss functions, these algorithms improve global structure preservation and run-time efficiency. Figure 13 demonstrates differences in results from DR methods on a dataset of isotropic Gaussians. As the figure shows, the results of different DR techniques look quite different from each other. Recent studies on DR algorithms shed light on how the loss function affects the rendering of local structure (Böhm et al., 2020), and provide guidance on how to design good loss functions so that the local and global structure can both be preserved simultaneously (Wang et al., 2020b). Nevertheless, several challenges still exists for DR methods:

7.1 How to capture information from the high dimensional space more accurately?

Most of the recent DR methods capture information in the high dimensional space mainly from the k-nearest-neighbors and their relative distances, at the expense of information from points that are more distant, which would allow the preservation of more global structure. Wattenberg et al. (2016); Coenen and Pearce (2019) discuss possible pitfalls in data analysis created by t-SNE and UMAP due to loss of non-local information. Recent methods mitigate the loss of global information by using global-aware initialization (McInnes et al., 2018; Kobak and Berens, 2019) (that is, initializing the distances in the low-dimensional space using PCA) and/or selectively preserving distances between non-neighbor samples (Fu et al., 2019; Wang et al., 2020b). Nevertheless, these methods are still designed and optimized under the assumption that the nearest neighbors, defined by the given metric (usually Euclidean distance in the high dimensional space), can accurately depict the relationships between samples. This assumption may not hold true for some data, for instance, Euclidean distance may not be suitable for measuring distances between weights (or activations) of a neural network (see Chan et al., 2019, for a detailed example of such a failure). We would like DR methods to better capture information from the high dimensional space to avoid such pitfalls.

7.2 How should we select hyperparameters for DR?

Modern DR methods, due to their multi-stage characteristics, involve a large number of hyperparameters, including the number of high-dimensional nearest neighbors to be preserved in the low-dimensional space and the learning rate used to optimize the low-dimensional embedding. There are often dozens of hyperparameters in any given DR method, and since DR methods are unsupervised and we do not already know the structure of the high-dimensional data, it is difficult to tune them. A poor choice of hyperparameters may lead to disappointing (or even misleading) DR results. Fig 14 shows some DR results for the Mammoth dataset (The Smithsonian Institute, 2020; Coenen and Pearce, 2019) using t-SNE, LargeVis, UMAP, TriMAP and PaCMAP with different sets of reasonable hyperparameters. When their perplexity parameter or the number of nearest neighbors is not chosen carefully, algorithms can fail to preserve the global structure of the mammoth (specifically, the overall placement of the mammoth's parts), and they create spurious clusters (losing connectivity between parts of the mammoth) and lose

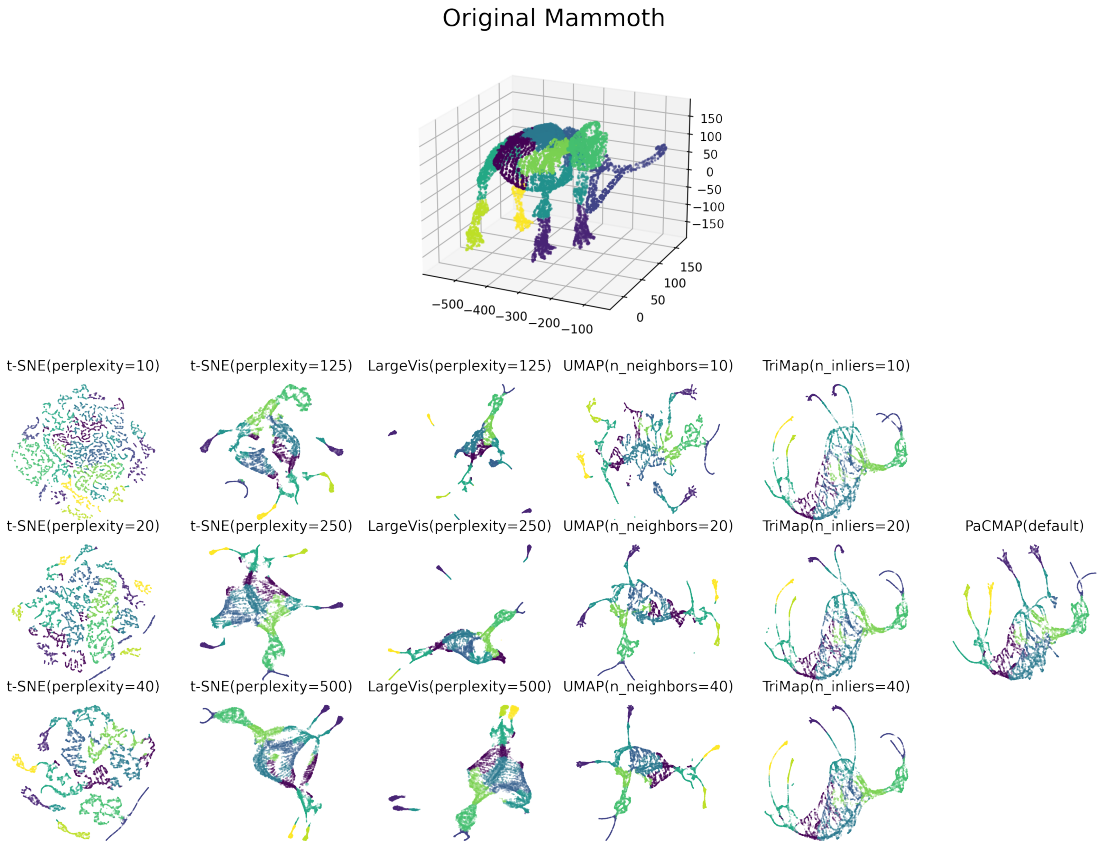


Figure 14: Projection from Wang et al. (2020b) of the Mammoth dataset into 2D using t-SNE (van der Maaten and Hinton, 2008; Linderman et al., 2019; Poličar et al., 2019), LargeVis (Tang et al., 2016), UMAP (McInnes et al., 2018), TriMap (Amid and Warmuth, 2019) and PaCMAP (Wang et al., 2020b). Incorrectly-chosen hyperparameters will lead to misleading results even in a simple dataset. This issue is particularly visible for t-SNE (first two columns) and UMAP (fourth column). The original dataset is 3 dimensional and is shown at the top.

details (such as the toes on the feet of the mammoth). For more detailed discussions about the effect of different hyperparameters, see Wattenberg et al. (2016); Coenen and Pearce (2019). Multiple works (for example Belkina et al., 2019; Wang et al., 2020b) aimed to alleviate this problem for the most influential hyperparameters, but the problem still exists, and the set of hyperparameters remains data-dependent. The tuning process, which sometimes involves many runs of a DR method, is time and power consuming, and requires user expertise in both the data domain and in DR algorithms. It could be extremely helpful to achieve better automatic hyperparameter selection for DR algorithms.

7.3 **Can the DR transformation from high- to low-dimensions be made more interpretable or explainable?** The DR mapping itself – that is, the transformation from high to low dimensions – typically is complex. There are some cases in which insight into this mapping can be gained, for instance, if PCA is used as the DR method, we may be able to determine which of the original dimensions are dominant in the first few principle components. It may be useful to

design modern approaches to help users understand how the final two or three dimensions are defined in terms of the high-dimensional features. This may take the form of explanatory post-hoc visualizations or constrained DR methods.

Example: Computational biologists often apply DR methods to single-cell RNA sequence data to understand the cell differentiation process and discover previously-unknown subtypes of cells. Without a suitable way to tune parameters, they may be misled by a DR method into thinking that a spurious cluster from a DR method is actually a new subtype of cell, when it is simply a failure of the DR method to capture local or global structure (Challenge 7.2 above). Since tuning hyperparameters in a high-dimensional space is difficult (without the ground truth afforded to supervised methods), the researchers have no way to see whether this cluster is present in the high-dimensional data or not (Challenge 7.1 above). Scientists could waste a lot of time examining each such spurious cluster. If we were able to solve the problems with DR tuning and structure preservation discussed above, it will make DR methods more reliable, leading to potentially increased understanding of many datasets.

8 Machine learning models that incorporate physics and other generative or causal constraints

There is a growing trend towards developing machine learning models that incorporate physics (or other) constraints. These models are not purely data-driven, in the sense that their training may require little data or no data at all (e.g., Rao et al., 2020). Instead, these models are trained to observe physical laws, often in the form of ordinary (ODEs) and partial differential equations (PDEs). These physics-guided models provide alternatives to traditional numerical methods (e.g., finite element methods) for solving PDEs, and are of immense interest to physicists, chemists, and materials scientists. The resulting models are interpretable, in the sense that they are constrained to follow the laws of physics that were provided to them. (It might be easier to think conversely: physicists might find that a standard supervised machine learning model that is trained on data from a known physical system – but that does not follow the laws of physics – would be uninterpretable.)

The idea of using machine learning models to approximate ODEs and PDEs solutions is not new. Lee and Kang (1990) developed highly parallel algorithms, based on neural networks, for solving finite difference equations (which are themselves approximations of original differential equations). Psichogios and Ungar (1992) created a hybrid neural network-first principles modeling scheme, in which neural networks are used to estimate parameters of differential equations. Lagaris et al. (1998); Lagaris et al. (2000) explored the idea of using neural networks to solve initial and boundary value problems. More recently, Raissi et al. (2019) extended the earlier works and developed the general framework of a *physics-informed neural network* (PINN). In general, a PINN is a neural network that approximates the solution of a set of PDEs with initial and boundary conditions. The training of a PINN minimizes the residuals from the PDEs as well as the residuals from the initial and boundary conditions. In general, physics-guided models (neural networks) can be trained without supervised training data. Let us explain how this works. Given a differential equation, say, $f'(t) = af(t) + bt + c$, where a, b and c are known constants, we could train a neural network g to approximate f, by minimizing $(g'(t) - ag(t) - bt - c)^2$ at finitely many

points t. Thus, no labeled data in the form $(t, f(t))$ (what we would need for conventional supervised machine learning) is needed. The derivative $g'(t)$ with respect to input t (at each of those finitely many points t used for training) is found by leveraging the existing network structure of g using back-propagation. Figure 15 illustrates the training process of a PINN for approximating the solution of one-dimensional heat equation $\frac{\partial u}{\partial t} = k\frac{\partial^2 u}{\partial x^2}$ with initial condition $u(x, 0) = f(x)$ and Dirichlet boundary conditions $u(0, t) = 0$ and $u(L, t) = 0$. If observed data are available, a PINN can be optimized with an additional mean-squared-error term to encourage data fit. Many extensions to PINNs have since been developed, including fractional PINNs (Pang et al., 2019a) and parareal PINNs (Meng et al., 2020). PINNs have been extended to convolutional (Zhu et al., 2019) and graph neural network (Seo and Liu, 2019) backbones. They have been used in many scientific applications, including fluid mechanics modeling (Raissi et al., 2020), cardiac activation mapping (Sahli Costabal et al., 2020), stochastic systems modeling (Yang and Perdikaris, 2019), and discovery of differential equations (Raissi et al., 2018; Raissi, 2018).

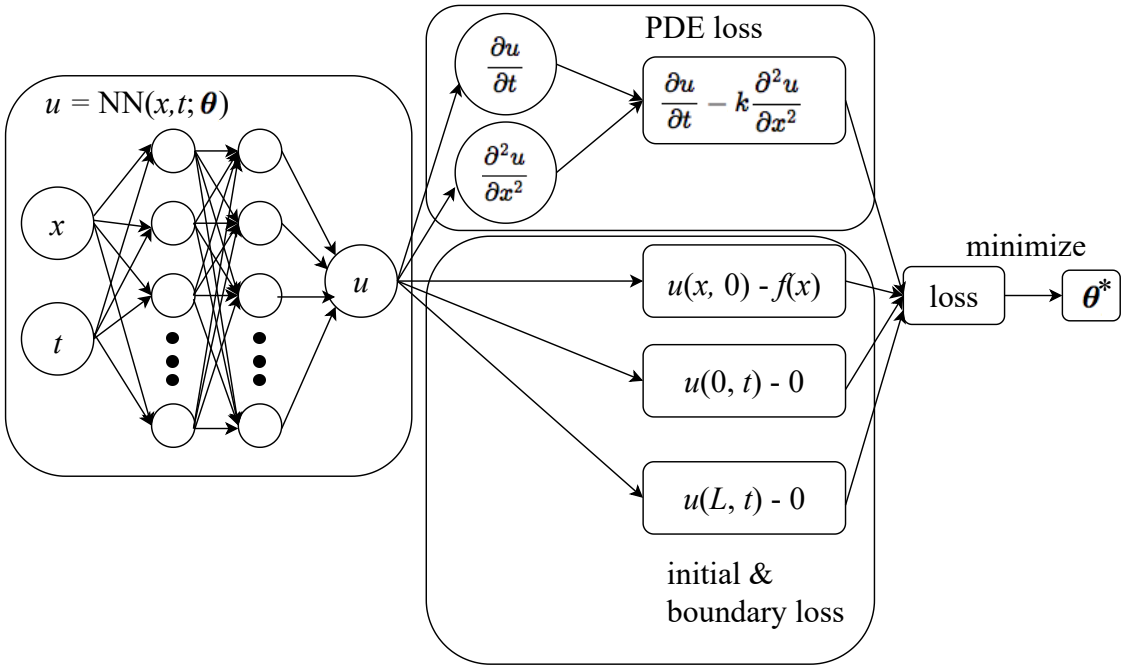


Figure 15: Physics-informed neural network (PINN) framework for solving the one-dimensional heat equation $\frac{\partial u}{\partial t} = k\frac{\partial^2 u}{\partial x^2}$ with initial condition $u(x, 0) = f(x)$ and Dirichlet boundary conditions $u(0, t) = 0$ and $u(L, t) = 0$. Here, u is the model, which is the output of a neural network (left), obeys the heat equation (upper right) and initial and boundary conditions (lower right). The degree to which u does not obey the heat equation or initial or boundary conditions is reflected in the loss function.

In addition to neural networks, Gaussian processes are also popular models for approximating solutions of differential equations. For example, Archambeau et al. (2007) developed a variational approximation scheme for estimating the posterior distribution of a system governed by a general stochastic differential equation, based on Gaussian processes. Zhao et al. (2011) developed a PDE-constrained Gaussian process model, based on the global Galerkin discretization of the governing PDEs for the wire saw slicing process. More recently, Pang et al. (2019b) used the neural-network-

induced Gaussian process (NNGP) regression for solving PDEs.

Despite recent success of PINNs or physics-guided machine learning in general, challenges still exist, including:

8.1 **How to improve training of a PINN?** Wang et al. (2020a) discussed the problem of numerical stiffness in gradient backpropagation during training of a PINN. To mitigate the problem, they proposed an algorithm to dynamically balance various loss terms during training. How to further improve training of a PINN is still an open challenge that needs to be addressed.

8.2 **How do we combine PINNs with optimal experimental/simulation design to quickly reduce uncertainty in approximating physical models?** Currently, data that might be used for training PINNs are often collected (from experiments or simulations) beforehand (i.e., prior to training of PINNs). This could lead to large uncertainty bands on areas of the input domain where data are scarce, and when the parameters of the PDE need to be learned from these data. Integrating experimental/simulation design with training of PINNs could lead to a more efficient data collection process and could reduce uncertainty of PINNs in approximating physical models. However, how to achieve this is still an open question.

8.3 **Are there other ways of incorporating PDE (or other) constraints into models other than neural networks?** Neural networks can be easily differentiated using the chain rule. This makes them ideal for approximating solutions to PDEs. Other models, especially non-parametric regression models cannot be differentiated easily – even though there have been some works that used Gaussian processes for approximating solutions to PDEs, these Gaussian-process solutions often rely on discrete (finite-difference) approximations for the original PDEs (see Zhao et al., 2011), or they could be non-differentiable at times where there are observations (see Archambeau et al., 2007). How one can incorporate differential constraints (in terms of differential equations) into non-parametric regression models remains an open question.

Going beyond PDEs to other types of complex constraints that encode prior knowledge, Kursuncu et al. (2020) proposed the concept of *knowledge infused* deep learning, where the learning of a deep neural network is guided by domain knowledge represented as a knowledge graph. Knowledge graphs encode relationships between entities in the world, including common sense knowledge. Thus, being able to incorporate information from a knowledge graph could be very helpful if we can figure out ways to do it effectively.

Example: Suppose that a team of aerospace engineers wants to study the aerodynamics for a new design of aircraft wing. They want to use a physics-guided machine learning model to approximate a solution to the physical model (a set of PDEs), because numerically computing the PDE solution is computationally expensive. They want to try different machine learning models, but they need to first think about how they can integrate the PDE constraints into machine learning models that are not easily differentiable with respect to input (Challenge 8.3). If they use a PINN, they need to ensure that training does not suffer from gradient pathologies (Challenge 8.1). In addition, collecting data to support training of a PINN by conducting experiments or running simulations can be expensive. They want to know how to integrate experimental/simulation design with training of PINNs to achieve the best model (Challenge 8.2). How to integrate constraints into various models,

and how to improve training and to co-design experiments/simulations with model training are challenges that burden not only aerospace engineers but any scientist working with physics-guided machine learning models.

9 Characterization of the "Rashomon" set of good models

In many practical machine learning problems, there is a multiplicity of almost-equally-accurate models. This set of high performing models is called the *Rashomon set*, based on an observation of the *Rashomon effect* by the statistician Leo Breiman. The Rashomon effect occurs when there are multiple descriptions of the same event (Breiman et al., 2001) with possibly no ground truth. The Rashomon set includes these multiple descriptions of the same dataset (and again, none of them are assumed to be the truth, see Semenova et al., 2019; Fisher et al., 2019; Dong and Rudin, 2020; Marx et al., 2020). Rashomon sets occur in multiple domains, including credit score estimation, medical imaging, natural language processing, health records analysis, recidivism prediction, and so on (D'Amour et al., 2020; Marx et al., 2020). We have discussed in Challenges 4 and 5 that even deep neural networks for computer vision exhibit Rashomon sets, because neural networks that perform case-based reasoning on disentanglement still yielded models that were equally accurate to their unconstrained values; thus, these interpretable deep neural models are within the Rashomon set. Rashomon sets present an opportunity for data scientists: if there are many equally-good models, we can choose one that has desired properties that go beyond minimizing an objective function. In fact, the model that optimizes the training loss might not be the best to deploy in practice anyway due to the possibilities of poor generalization, trust issues, or encoded inductive biases that are undesirable (D'Amour et al., 2020). More careful approaches to problem formulation and model selection could be taken that include the possibility of model multiplicity in the first place. Simply put – we need ways to explore the Rashomon set, particularly if we are interested in model interpretability.

Formally, the Rashomon set is the set of models whose training loss is below a specific threshold, as shown in Figure 16 (a). Given a loss function and a model class $\mathcal{F}$, the Rashomon set can be written as

$$R(\mathcal{F}, f^*, \epsilon) = \{f \in \mathcal{F} \text{ such that } Loss(f) \leq Loss(f^*) + \epsilon\},$$

where f^* can be an empirical risk minimizer, optimal model or any other reference model. We would typically choose $\mathcal{F}$ to be complex enough to contain models that fit the training data well without overfitting. The threshold ϵ, which is called the Rashomon parameter, can be a hard hyper-parameter set by a machine learning practitioner or a percentage of the loss (i.e., ϵ becomes $\epsilon' Loss(f^*)$). We would typically choose ϵ or ϵ' to be small enough that suffering this additional loss would have little to no practical significance on predictive performance. For instance, we might choose it to be much smaller than the (generalization) error between training and test sets. We would conversely want to choose ϵ or ϵ' to be as large as permitted so that we have more flexibility to choose models within a bigger Rashomon set.

It has been shown by Semenova et al. (2019) that when the Rashomon set is large, under weak assumptions, it must contain a simple (perhaps more interpretable) model within it. The argument goes as follows: assume the Rashomon set is large, so that it contains a ball of functions from a complicated function class $\mathcal{F}_{\text{complicated}}$ (think of this as high-dimensional polynomials that are complex enough to fit the data well without overfitting). If a set of simpler functions $\mathcal{F}_{\text{simpler}}$ could

serve as approximating set for $\mathcal{F}_{\text{complicated}}$ (think decision trees of a certain depth approximating the set of polynomials), it means that each complicated function could be well-approximated by a simpler function (and indeed, polynomials can be well-approximated by decision trees). By this logic, the ball of $\mathcal{F}_{\text{complicated}}$ that is within the Rashomon set must contain at least one function within $\mathcal{F}_{\text{simpler}}$, which is the simple function we were looking for.

Semenova et al. (2019) also suggested a useful rule of thumb for determining whether a Rashomon set is large: run many different types of machine learning algorithms (e.g., boosted decision trees, support vector machines, neural networks, random forests, logistic regression) and if they generally perform similarly, it correlates with the existence of a large Rashomon set (and thus the possibility of a simpler function also achieving a similar level of accuracy). The knowledge that there might exist a simple-yet-accurate function before finding it could be very useful, particularly in the cases where finding an optimal sparse model is NP-hard, as in Challenge 1. Here, the user would run many different algorithms to determine whether it would be worthwhile to solve the hard problem of finding an interpretable sparse model.

The knowledge of the functions that lie within Rashomon sets sees value in multiple interesting use cases. Models with various important properties besides interpretability can exist in the Rashomon set, including fairness (Coston et al., 2021) and monotonicity. Thus, understanding the properties of the Rashomon set could be pivotal for analysis of a complex machine learning problem and its possible modeling choices.

The *size of the Rashomon set* can be considered as a way of measuring the *complexity of a learning problem*. Problems with large Rashomon sets are less complicated problems, since more good solutions exist to these problems. The Rashomon set is a property of the model class and dataset. The size of the Rashomon set differs from other known characterizations of the complexity in machine learning. Ways that complexity of function classes, algorithms, or learning problems are typically measured in statistical learning theory include the VC dimension, covering numbers, algorithmic stability, Rademacher complexity, or flat minima (see, for instance, Srebro et al., 2010; Kakade et al., 2009; Zhou, 2002; Schapire et al., 1998; Koltchinskii and Panchenko, 2002; Bousquet and Elisseeff, 2002; Vapnik and Chervonenkis, 1971); the size of the Rashomon set differs from all of these quantities in fundamental ways, and it is important in its own right for showing the existence of simpler models.

A useful way to represent the hypothesis for a problem is by projecting it to variable importance space, where each axis represents the importance of a variable. That way, a single function is represented by a point in this space (i.e., a vector of coordinates), where each coordinate represents how important a variable is to that model. (Here, variable importance for variable v is measured by *model reliance* or *conditional model reliance*, which represents the change in loss we would incur if we scrambled variable v.) The Rashomon set can be represented as a subset of this variable importance space. Fisher et al. (2019) used this representation to create a *model independent* notion of variable importance: specifically, they suggest to consider the maximum and minimum of variable importance across all models in the Rashomon set. This is called the Model Class Reliance (MCR). For any user who would pick a "good" model (i.e., a model in the Rashomon set), then the importance of the variable is within the range given by the MCR. Dong and Rudin (2020) expand on this by suggesting to visualize the "cloud" of variable importance values for models within the Rashomon set. This cloud helps us understand the Rashomon set well enough to choose a model we might find interpretable.

Current research works have barely scratched the characterization and usage of model multi-

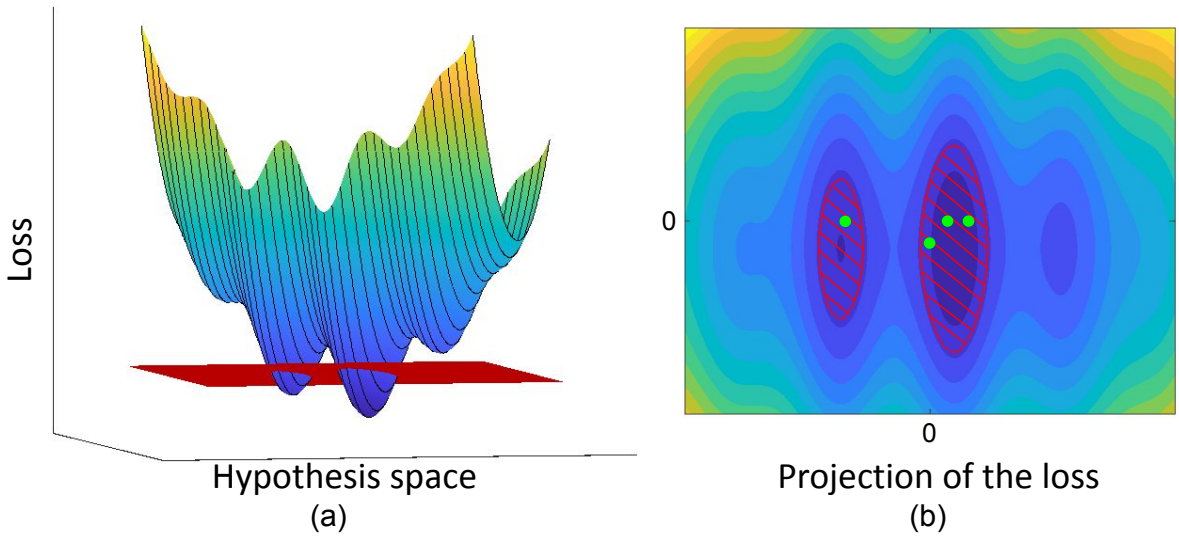


Figure 16: (a) An illustration of a possible Rashomon set in two-dimensional hypothesis space. Models from two local minima that are below the red plane belong to the Rashomon set. (b) An illustration of a possible visualization of the Rashomon set. Models inside the shaded red regions belong to the Rashomon set. The plot is created as a contour plot of the loss over the hypothesis space. Green dots represent a few simpler models inside the Rashomon set. These models are simpler because they are sparse: they depend on one less dimension than other models in the Rashomon set.

plicity and Rashomon sets in machine learning. Exploring the Rashomon set is critical if we hope to find interpretable models within the set of accurate models. We ask the following questions about the Rashomon set:

9.1 **How can we characterize the Rashomon set?** As discussed above, the volume of the Rashomon set is a useful indirect indicator of the existence of interpretable models. There have been several works that suggest computing statistics of the Rashomon set in parameter space, such as the volume in parameter space, which is called the *Rashomon volume*. The Rashomon volume or other statistics that characterize the Rashomon set can be computed more easily if the volume in parameter space has a single global minimum (Hochreiter and Schmidhuber, 1997; Dinh et al., 2017; Keskar et al., 2016; Chaudhari et al., 2016; Keskar et al., 2016; Chaudhari et al., 2019); these variations include ϵ-flatness (Hochreiter and Schmidhuber, 1997; Dinh et al., 2017) and ϵ-sharpness (Keskar et al., 2016; Dinh et al., 2017). The *Rashomon ratio* provides some perspective on the size of the Rashomon set, defined as the ratio of the Rashomon volume to the volume of the whole hypothesis space (Semenova et al., 2019), assuming that both are bounded.

There are several problems that arise when computing the size of the Rashomon set in parameter space. First, the volume of a set of functions can be hard to compute or estimate, but there are ways to do it, including sampling (or analytical computation in some simple cases when there is one minimum). Second, the choice of parameterization of the hypothesis space matters when working in parameter space. Overparametrization or underparametrization can cause volumes in parameter space to be artificially made larger or smaller without a change

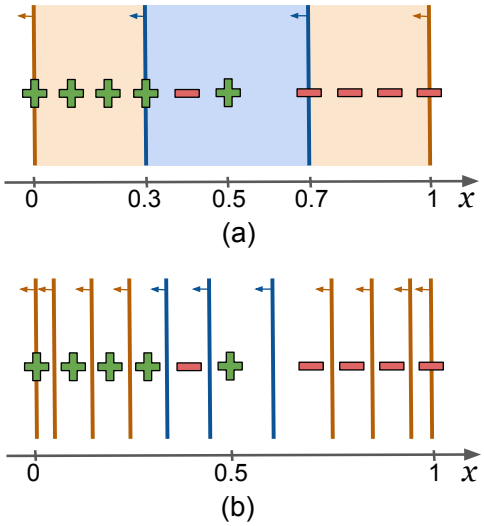


Figure 17: A one-dimensional example of computation of the Rashomon ratio. The hypothesis space consists of decision stumps $f(x) = \{\mathbb{1}_{[x \leq a]}\}, a \in [0, 1]$. (The function is 1 if $x \leq a$ and 0 otherwise.) The loss function is the zero-one loss: $loss(f, x, y) = \mathbb{1}_{[f(x)=y]}$, which is 1 if the model makes a mistake and zero otherwise. A model f belongs to the Rashomon set if $\sum_i loss(f, x_i, y_i) \leq 0.2$, that is, the model made 2 mistakes. The Rashomon ratio in (a) is equal to 0.4 and is computed as the ratio of volumes of decision stumps in the Rashomon set (blue shaded region) to the volume of all possible decision stumps in the area where data reside (blue and orange shaded region). The pattern Rashomon ratio in (b) is equal to $3/11$ and is computed as the ratio of classifiers that belong to the Rashomon set (blue stumps, of which there are 3 unique stumps with respect to the data) to the number of all possible classifiers (orange and blue stumps, of which there are 11). This figure shows that there are multiple ways to compute the size of the Rashomon set and it is not clear which one to choose.

in the actual space of functions (Dinh et al., 2017). For instance, if we make a copy of a variable and include it in the datasets, it can change the Rashomon ratio. A third but separate problem with the Rashomon ratio is its denominator: the denominator is the volume of all models we might reasonably consider before seeing the data. If we change the set of models we might consider, we change the denominator, and the ratio changes. And, because different model spaces are different, there might not be a way to directly compare the Rashomon ratios computed on different hypothesis spaces to each other.

This brings us to another way to measure the Rashomon set: The *pattern Rashomon ratio* (Semenova et al., 2019; Marx et al., 2020), which could potentially be used to handle some of these problems. The pattern Rashomon ratio considers unique predictions on the data (called "patterns") rather than the count of functions themselves. In other words, the pattern Rashomon ratio considers the count of predictions that could be made rather than a count of functions. Figure 17 illustrates the difference between the Rashomon ratio and the pattern Rashomon ratio. The pattern Rashomon ratio has a fixed denominator (all possible patterns

one could produce on the data using functions from the class). A direct comparison of the pattern Rashomon ratio over function classes is meaningful. However, it is still difficult to compute. It is also more sensitive to small shifts and changes in data.

Despite that multiple measures, including ϵ-flatness, ϵ-sharpness, Rashomon ratio, and pattern ratio, have been proposed to measure the size of the Rashomon set, they do not provide a universal and simple-to-calculate way of characterization the size of the Rashomon set. Many challenges to remain open:

(a) **What is a good space in which to measure the size of the Rashomon set? What is a good measure of the size of the Rashomon set?** As discussed above, the parameter space suffers from issues with parameterization, but the pattern Rashomon ratio has its own problems. Would variable importance space be suitable, where each axis represents the importance of a raw variable, as in the work of Dong and Rudin (2020)? Or is there an easier space and metric to work with?

(b) **Are there approximation algorithms or other techniques that will allow us to efficiently compute or approximate the size of the Rashomon set?** At worst, computation over the Rashomon set requires a brute force calculation over a large discrete hypothesis space. In most cases, the computation should be much easier. Perhaps we could use dynamic programming or branch and bound techniques to reduce the search space so that it encompasses the Rashomon set but not too much more than that? In some cases, we could actually compute the size of the Rashomon set analytically. For instance, for linear regression, a closed-form solution in parameter space for the volume of the Rashomon set has been derived based on the singular values of the data matrix (Semenova et al., 2019).

9.2 **What techniques can be used to visualize the Rashomon set?** Visualization of the Rashomon set can potentially help us to understand its properties, issues with the data, biases, or underspecification of the problem. To give an example of how visualization can be helpful in troubleshooting models generally, Li et al. (2017) visualized the loss landscape of neural networks and, as a result, found answers to questions about the selection of the batch size, optimizer, and network architecture. Kissel and Mentch (2021) proposed a tree-based graphical visualization to display outputs of the *model path selection procedure* that finds models from the Rashomon set based on a forward selection of variables. The visualization helps us to understand the stability of the model selection process, as well as the richness of the Rashomon set, since wider graphical trees imply that there are more models available for the selection procedure. To characterize the Rashomon set in variable importance space, *variable importance diagrams* have been proposed (Dong and Rudin, 2020), which are 2-dimensional projections of the *variable importance cloud*. The *variable importance cloud* is created by mapping every variable to its importance for every good predictive model (every model in the Rashomon set). Figure 16(b) uses a similar technique of projection in two-dimensional space and depicts the visualization of the example of the Rashomon set of Figure 16(a). This simple visualization allows us to see the Rashomon set's layout, estimate its size or locate sparser models within it. Can more sophisticated approaches be developed that would allow good visualization? The success of these techniques most likely will have to depend on whether we can design a good

metric for the model class. After the metric is designed, perhaps we might be able to utilize techniques from Challenge 7 for the visualization.

9.3 **What model to choose from the Rashomon set?** When the Rashomon set is large, it can contain multiple accurate models with different properties. Choosing between them might be difficult, particularly if we do not know how to explore the Rashomon set. Interactive methods might rely on dimension reduction techniques (that allow users to change the location of data on the plot or to change the axis of visualization), weighted linear models (that allow the user to compute weights on specific data points), continuous feedback from the user (that helps to continuously improve models prediction in changing environments, for example, in recommender systems) in order to interpret or choose a specific model with desired property. Das et al. (2019) design a system called BEAMES that allows users to interactively select important features, change weights on data points, visualize and select a specific model or even an ensemble of models. BEAMES searches the hypothesis space for models that are close to the practitioners' constraints and design choices. The main limitation of BEAMES is that it works with linear regression classifiers only. Can a similar framework that searches the Rashomon set, instead of the whole hypothesis space, be designed? What would the interactive specification of constraints look like in practice to help the user choose the right model? Can collaboration with domain experts in other ways be useful to explore the Rashomon set?

Example: Suppose that a financial institution would like to make data-driven loan decisions. The institution must have a model that is as accurate as possible, and must provide reasons for loan denials. In practice, loan decision prediction problems have large Rashomon sets, and many machine learning methods perform similarly despite their different levels of complexity. To check whether the Rashomon set is indeed large, the financial institution wishes to measure the size of the Rashomon set (Challenge 9.1) or visualize the layout of it (Challenge 9.2) to understand how many accurate models exist, and how many of them are interpretable. If the Rashomon set contains multiple interpretable models, the financial institution might wish to use an interactive framework (Challenge 9.3) to navigate the Rashomon set and design constraints that would help to locate the best model for their purposes. For example, the institution could additionally optimize for fairness or sparsity.

10 Interpretable reinforcement learning

Reinforcement learning (RL) methods determine what actions to take at different states in order to maximize a cumulative numerical reward (e.g., see Sutton and Barto, 2018). In RL, a learning agent interacts with the environment by a trial and error process, receiving signals (rewards) for the actions it takes. Recently, deep reinforcement learning algorithms have achieved state-of-the-art performance in mastering the game of Go and various Atari games (Silver et al., 2016; Mnih et al., 2013) and have been actively used in robotic control (Kober et al., 2013; Tai et al., 2016), simulated autonomous navigation and self-driving cars (Kiran et al., 2021; Sallab et al., 2017), dialogue summarization, and question answering (Li et al., 2016). RL has also been applied to high-stakes decisions, such as healthcare and personalized medicine, including approaches for dynamic treatment regimes (Liu et al., 2017; Yu et al., 2019), treatment of sepsis (Komorowski

et al., 2018), treatment of chronic disorders including epilepsy (Guez et al., 2008), for HIV therapy (Parbhoo et al., 2017), and management of Type 1 diabetes (Javad et al., 2019).

In reinforcement learning, at each timestep, an agent observes the state (situation) that it is currently in, chooses an action according to a policy (a learned mapping from states to actions), and receives an intermediate reward that depends on the chosen action and the current state, which leads the agent to the next state, where the process repeats. For example, let us consider an RL system that is used for the management of glucose levels of Type 1 diabetes patients by regulating the amount of synthetic insulin in the blood. The state is represented by the patient's information, including diet, exercise level, weight, so on; actions include the amount of insulin to inject; rewards are positive if the blood sugar level is within a healthy range and negative otherwise. Figure 18 shows the schematic diagram of a reinforcement learning system based on the diabetes example.

Reinforcement learning is more challenging than a standard supervised setting: the agent does not receive all the data at once (in fact, the agent is responsible for collecting data by exploring the environment); sometimes besides immediate rewards, there are delayed rewards when feedback is provided only after some sequence of actions, or even after the goal is achieved (for example, the positive reward on a self-driving car agent is provided after it reaches a destination). Because reinforcement learning involves long-term consequences to actions, because actions and states depend on each other, and because agents need to collect data through exploration, interpretability generally becomes more difficult than in a standard supervised learning. Search spaces for RL can be massive since we need to understand which action to take in each state based on an estimate of future rewards, uncertainty of these estimates, and exploration strategy. In deep reinforcement learning (Mnih et al., 2013; Silver et al., 2016), policies are defined by deep neural networks, which helps to solve complex applied problems, but typically the policies are essentially impossible to understand or trust.

Interpretability could be particularly valuable in RL. Interpretability might help to reduce the RL search space; if we understand the choices and intent of the RL system, we can remove actions that might lead to harm. Interpretability could make it easier to troubleshoot (and use) RL for long-term medical treatment or to train an agent for human-robot collaboration.

One natural way to include interpretability in a reinforcement learning system is by representing the policy or value function by a decision tree or rule list (Roth et al., 2019; Silva et al., 2020). In a policy tree representation, the nodes might contain a small set of logical conditions, and leaves might contain an estimate of the optimal probability distribution over actions (see Figure 19). A policy tree can be grown, for example, in a greedy incremental way by updating the tree only when the estimated discounted future reward would increase sufficiently (Roth et al., 2019); in the future, one could adapt methods like those from Challenge 1 to build such trees.

Other methods make specific assumptions about the domain or problem that allows for interpretable policies. One of the most popular simplifications is to consider a symbolic or relational state representation. In this case, the state is a combination of logical or relational statements. For example, in the classic Blocks World domain, where the objective is to learn how to stack blocks on top of each other to achieve a target configuration, these statements comprising the state might look like *on(block1, block2)*; *on(block1, table)*; *free(block2)*. Typically, in relational reinforcement learning, policies are represented as relational regression trees (Džeroski et al., 1998, 2001; Kersting and Driessens, 2008; Finney et al., 2012; Das et al., 2020) that are more interpretable than neural networks.

A second set of assumptions is based on a natural decomposition of the problem for multi-task

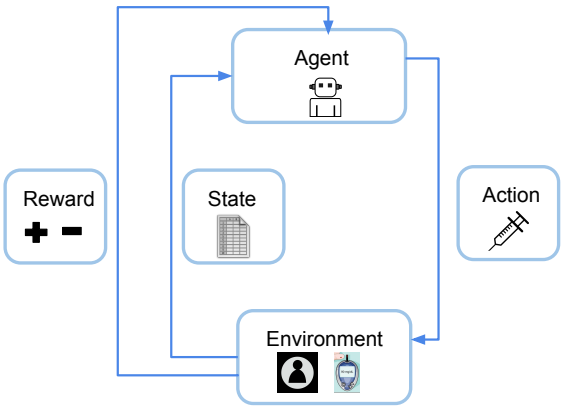


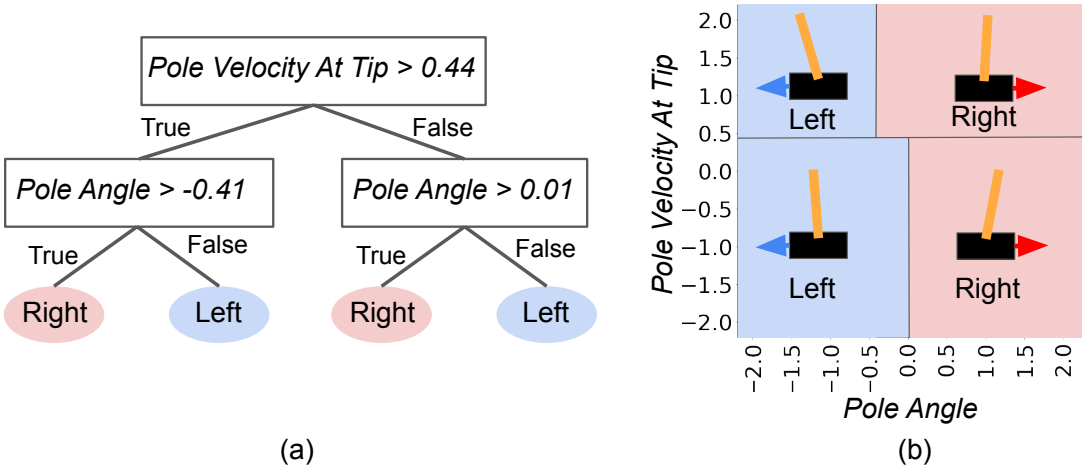
Figure 18: An illustration of an RL system based on the example of Type 1 diabetes management. This is theoretically a closed-loop system, where the agent observes the state from the environment, makes an action, and receives a reward based on the action. For diabetes control, the state consists of measurements of glucose and other patient data, the actions are injections of insulin, and the rewards are based on sugar levels.

or skills learning. For example, Shu et al. (2017) use hierarchical policies that are learned over multiple tasks, where each task is decomposed into multiple sub-tasks (skills). The description of the skills is created by a human, so that the agent learns these understandable skills (for instance, task *stack blue block* can be decomposed into the skills *find blue block*, *get blue block*, and *put blue block*).

As far as we know, there are currently no general interpretable well-performing methods for deep reinforcement learning that allow transparency in the agent's actions or intent. Progress has been made instead on explainable deep reinforcement learning (posthoc approximations), including tree-based explanations (Coppens et al., 2019; Liu et al., 2018; Dhebar et al., 2020; Bastani et al., 2018; Dahlin et al., 2020), reward decomposition (Juozapaitis et al., 2019), and attention-based methods (Zambaldi et al., 2018; Mott et al., 2019). An interesting approach is that of Verma et al. (2018), who define rule-based policies through a domain-specific, human-readable programming language that generalizes to unseen environments, but shows slightly worse performance than neural network policies it was designed to explain. The approach works for deterministic policies and symbolic domains only and will not work for the domains where the state is represented as a raw data image, unless an additional logical relation extractor is provided.

Based on experience from supervised learning (discussed above), we know that post-hoc explanations suffer from multiple problems, in that explanations are often incorrect or incomplete. Atrey et al. (2020) have argued that for deep reinforcement learning, saliency maps should be used for exploratory, and not explanatory, purposes. Therefore, developing interpretable reinforcement learning policies and other interpretable RL models is important. The most crucial challenges for interpretable reinforcement learning area include:

10.1 **What constraints would lead to an accurate interpretable policy?** In relational methods, policies or value functions could be represented by decision or regression trees, however, since these methods work for symbolic domains only, they have limited practical use without a method of extracting interpretable relations from the domain. Can interpretable deep neural



(a) (b)

Figure 19: (a) An example of a learned decision tree policy for the Cart-Pole balancing problem adapted from Silva et al. (2020). In the Cart-Pole environment, a pole is attached to a cart by a joint. The cart moves along the horizontal axis. The goal is to keep the pole balanced upright, preventing it from falling. The state consists of four features: *Cart Position, Cart Velocity, Pole Angle and Pole Velocity At Tip*. Here, angle 0° means straight upwards. There are two possible actions at each state, including *push cart to the left* or *push cart to the right*. (b) is a visualization of the policy from Silva et al. (2020) based on two features: *Pole Angle* and *Pole Velocity At Tip*. Interestingly, it is easy to see from this figure that the policy is not left-right symmetric. Note that there are many reasonable policies for this problem, including many that are asymmetric.

network methods similar to those in Challenge 5 provide an interpretable policy without the performance compromise? Are there other ways of adding constraints or structure to the policy that will ensure human-understandable actions and intent? Will adding other sparsity or interpretability constraints, as in Challenges 1 and 2, to the policy or value function help improve interpretability without sacrificing accuracy? Can we design interpretable models whose explanations can capture longer term consequences of actions?

10.2 **Under what assumptions does there exist an interpretable policy that is as accurate as the more complicated black-box model?** Reinforcement learning systems are tricky to train, understand and debug. It is a challenge in itself to decide what constraints will lead to an interpretable policy without loss of performance. For example, for PIRL (Verma et al., 2018), there was a drop in performance of the resulting rule-based policy. Of course, one of the possible reasons for this drop could be the fact that it used posthoc approximations rather than inherently interpretable models; another reason is that there might not be a well-performing sparse rule-based policy. So, the question is: how would we know if it is useful to optimize for an interpretable policy? A simple test that tells the modeler if such a model exists (similar to the characterizations of the Rashomon sets in Challenge 9) would be helpful before trying to develop an algorithm to find this well-performing model (if it exists).

10.3 **Can we simplify or interpret the state space?** Deep reinforcement learning often operates on complex data (raw data) or huge, if not continuous state spaces. The agent might be re-

quired to do a lot of exploration to learn useful policies in this case. Simplifying the state space might not only make the RL system learn more efficiently, but might also help humans to understand the decisions it is making. For example, by applying t-SNE to the last layer activation data of Mnih et al. (2013)'s deep Q-learning model, Zahavy et al. (2016) showed that the network automatically discovers hierarchical structures. Previously, these structures had not been used to simplify the state space. Zhang et al. (2021) improve over batch RL by computing the policy only at specific decision points where clinicians treat patients differently, instead of at every timestep. This resulted in a significantly smaller state space and faster planning.

Example: Consider a machine learning practitioner that is working on planning for assistive robots that help the elderly to perform day-to-day duties. In cases when the robot does some damage during its task execution or chooses a unusual plan to execute, it might be hard to understand the logic behind the robot's choices. If the practitioner can check that an interpretable model exists (Challenge 10.2), optimize for it (Challenge 10.1), and interpret the state space (Challenge 10.3), debugging the robot's failed behavior and correcting it can be much faster than without these techniques. Further, interpretability will help to strengthen trust between the robot and the human its assisting, and help it to generalize to new unforeseen cases.

11 Problems that weren't in our top 10 but are really important

We covered a lot of ground in the 10 challenges above, but we certainly did not cover all of the important topics related to interpretable ML. Here are a few that we left out:

- **Can we improve preprocessing to help with both interpretability and test accuracy?** As we discussed, some interpretable modeling problems are computationally difficult (or could tend to overfit) without preprocessing. For supervised problems with tabular data, popular methods like Principal Component Analysis (PCA) would generally transform the data in a way that damages interpretability of the model. This is because each transformed feature is a combination of all of the original features. Perhaps there are other general preprocessing tools that would preserve predictive power (or improve it), yet retain interpretability.

- **Can we convey uncertainty clearly?** Uncertainty quantification is always important. Tomsett et al. (2020) discusses the importance of uncertainty quantification and interpretability, and Antoràn et al. (2020) discusses interpretable uncertainty estimation.

- **Can we divide the observations into easier cases and harder cases, assigning more interpretable models to the easier cases?** Not all observations are equally easy to classify. As noted in the work of Wang (2019); Wang et al. (2021b), it is possible that easier cases could be handled by simpler models, leaving only the harder cases for models with less interpretability.

- **Do we need interpretable neural networks for tabular data?** There have been numerous attempts to design interpretable neural networks for tabular data, imposing sparsity, monotonicity, etc. However, it is unclear whether there is motivation for such formulations, since

neural networks do not generally provide much benefit for tabular data over other kinds of models. One nice benefit of the "optimal" methods discussed above is that when we train them we know how far we are from a globally optimal solution; the same is not true for neural networks, rendering them not just less interpretable but also harder to implement reliably.

- **What are good ways to co-design models with their visualizations?** Work in data and model visualization will be important in conveying information to humans. Ideally, we would like to co-design our model in a way that lends itself to easier visualization. One particularly impressive interactive visualization was done by Gomez et al. (2020) for a model on loan decisions. Another example of an interactive model visualization tool is that of Chen et al. (2021a). These works demonstrate that as long as the model can be visualized effectively, it can be non-sparse but still be interpretable. In fact, these works on loan decisions constitute an example of when a sparse model might not be interpretable, since people might find a loan decision model unacceptable if it does not include many types of information about the applicant's credit history. Visualizations can allow the user to hone in on the important pieces of the model for each prediction.

- **Can we do interpretable matching in observational causal inference?** In the classical potential outcomes framework of causal inference, matching algorithms are often used to match treatment units to control units. Treatment effects can be estimated using the matched groups. Matching in causal inference can be considered a form of case-based reasoning (Challenge 4). Ideally, units within each matched group would be similar on important variables. One such framework for matching is the Almost Matching Exactly framework (also called "learning to match," see Wang et al., 2021a), which learns the important variables using a training set, and prioritizes matching on those important variables.

  Matching opens the door for interpretable machine learning in causal inference because it produces treatment and control data whose distributions overlap. Interpretable machine learning algorithms for supervised classification discussed in Challenges 1, 2, and 3 can be directly applied to matched data; for instance, a sparse decision tree can be used on matched data to obtain an interpretable model for individual treatment effects, or an interpretable policy (i.e., treatment regimes).

- **How should we measure variable importance?** There are several types of variable importance measures: those that measure how important a variable is to a specific prediction, those that measure how important a variable is generally to a model, and those that measure how important a variable is independently of a model. Inherently interpretable models should generally not need the first one, as it should be clear how much a variable contributed to the prediction; for instance, for decision trees and other logical models, scoring systems, GAMs, case-based reasoning models, and disentangled neural networks, the reasoning process that comes with each prediction tells us explicitly how a variable (or concept) contributed to a given prediction. There are many explainable ML tutorials to describe how to estimate the importance of variables for black box models. For the other two types of variable importance, we refer readers to the work of Fisher et al. (2019), which has relevant references. If we can decide on an ideal variable importance measure, we may be able to create interpretable models with specific preferences on variable importance.

- **What are the practical challenges in deploying interpretable ML models?** There are many papers that provide useful background about the use of interpretability and explainability in practice. For instance, an interesting perspective is provided by Bhatt et al. (2020) who conduct an explainability survey showing that explanations often used only internally for troubleshooting rather than being shown to users. Kaur et al. (2020); Poursabzi-Sangdeh et al. (2018) (among others) have performed extensive human experiments on interpretable models and posthoc explanations of black box models, with some interesting and sometimes nonintuitive results.

- **What type of explanation is required by law?** The question of what explanations are legally required is important and interesting, but we will leave those to legal scholars such as Bibal et al. (2020); Wachter et al. (2017). Interestingly, scholars have explicitly stated that a "Right to Explanation" for automated decision-making does not actually exist in the European Union's General Data Protection Regulation, despite the intention (Wachter et al., 2017). Rudin (2019) proposes that for a high-stakes decision that deeply affects lives, no black box model should be used unless the deployer can prove that no interpretable model exists with a similar accuracy. If enacted, this would likely mean that black box models rarely (if ever) would be deployed for high-stakes decisions.

- **What are other forms of interpretable models?** It is not possible to truly review all interpretable ML models. One could argue that most of applied Bayesian statistics fits into our definition of interpretable ML because the models are constrained to be formed through an interpretable generative process. This is indeed a huge field, and numerous topics that we were unable to cover also would have deserved a spot in this survey.

12 Conclusion

In this survey, we hoped to provide a pathway for readers into important topics in interpretable machine learning. The literature currently being generated on interpretable and explainable AI can be downright confusing. The sheer diversity of individuals weighing in on this field includes not just statisticians and computer scientists but legal experts, philosophers, and graduate students, many of whom have not either built or deployed a machine learning model ever. It is easy to underestimate how difficult it is to convince someone to use a machine learning model in practice, and interpretability is a key factor. Many works over the last few years have contributed new terminology, mistakenly subsumed the older field of interpretable machine learning into the new field of "XAI," and review papers have universally failed even to truly distinguish between the basic concepts of explaining a black box and designing an interpretable model. Because of the misleading terminology, where papers titled "explainability" are sometimes about "interpretability" and vice versa, it is very difficult to follow the literature (even for us). At the very least, we hoped to introduce some fundamental principles, and cover several important areas of the field and show how they relate to each other and to real problems. Clearly this is a massive field that we cannot truly hope to cover, but we hope that the diverse areas we covered and problems we posed might be useful to those needing an entrance point into this maze.

Interpretable models are not just important for society, they are also beautiful. One might also find it absolutely magical that simple-yet-accurate models exist for so many real-world datasets.

We hope this document allows you to see not only the importance of this topic but also the elegance of its mathematics and the beauty of its models.

Acknowledgments

We thank Leonardo Lucio Custode for pointing out several useful references to Challenge 10. Thank you to David Page for providing useful references on early explainable ML. Thank you to the anonymous reviewers that made extremely helpful comments.

References

Agnar Aamodt and Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 7(1):39–59, 1994.

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In Proceedings of Neural Information Processing Systems (NeurIPS), 2018.

Tameem Adel, Zoubin Ghahramani, and Adrian Weller. Discovering interpretable representations for both deep generative and discriminative models. In Proceedings of International Conference on Machine Learning (ICML), pages 50–59, 2018.

Michael Anis Mihdi Afnan, Cynthia Rudin, Vince Conitzer, Julian Savulescu, Abhishek Mishra, Yanhe Liu, and Masoud Afnan. Ethical implementation of artificial intelligence to select embryos in *In Vitro Fertilization*. In Proceedings of the Fourth AAAI/ACM Conference on Artificial Intelligence, Ethics, and Society (AIES), 2021.

Rishabh Agarwal, Nicholas Frosst, Xuezhou Zhang, Rich Caruana, and Geoffrey E Hinton. Neural additive models: Interpretable machine learning with neural nets. arXiv preprint arXiv:2004.13912, 2020.

Sina Aghaei, Andres Gomez, and Phebe Vayanos. Learning optimal classification trees: Strong max-flow formulations. arXiv e-print arXiv:2002.09142, 2020.

D. Agrawal, P. Bernstein, E. Bertino, S. Davidson, U. Dayal, M. Franklin, and J. Widom. Challenges and opportunities with big data: A white paper prepared for the computing community consortium committee of the computing research association. Technical report, CRA, 2012. URL http://cra.org/ccc/resources/ccc-led-whitepapers/.

Ulrich Aïvodji, Julien Ferry, Sébastien Gambs, Marie-José Huguet, and Mohamed Siala. Learning fair rule lists. arXiv e-print arXiv:1909.03977, 2019.

E. Amid and M. K. Warmuth. TriMAP: Large-scale Dimensionality Reduction Using Triplets. arXiv e-print arXiv:1910.00204, 2019.

E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, and C. Rudin. Learning certifiably optimal rule lists for categorical data. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2017.

Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Certifiably optimal rule lists for categorical data. Journal of Machine Learning Research, 18:1–78, 2018.

Plamen Angelov and Eduardo Soares. Towards explainable deep neural networks (xDNN). Neural Networks, 130:185–194, 2020.

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias. Available from: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, May 2016.

Elliott M Antman, Marc Cohen, Peter JLM Bernink, Carolyn H McCabe, Thomas Horacek, Gary Papuchis, Branco Mautner, Ramon Corbalan, David Radley, and Eugene Braunwald. The TIMI risk score for unstable angina/non–ST elevation MI: a method for prognostication and therapeutic decision making. Journal of the American Medical Association, 284(7):835–842, 2000.

Javier Antoràn, Umang Bhatt, Tameem Adel, Adrian Weller, and José Miguel Hernández-Lobato. Getting a clue: A method for explaining uncertainty estimates, 2020.

Cedric Archambeau, Dan Cornford, Manfred Opper, and John Shawe-Taylor. Gaussian process approximations of stochastic differential equations. In Gaussian Processes in Practice, pages 1–16. PMLR, 2007.

Maryam Ashoori and Justin D. Weisz. In AI we trust? factors that influence trustworthiness of AI-infused decision-making processes. arXiv e-print arXiv:1912.02675, 2019.

Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. In International Conference on Learning Representations (ICLR), 2020.

Marcus A. Badgeley, John R. Zech, Luke Oakden-Rayner, Benjamin S. Glicksberg, Manway Liu, William Gale, Michael V. McConnell, Bethany Percha, Thomas M. Snyder, and Joel T. Dudley. Deep learning predicts hip fracture using confounding patient and healthcare variables. npj Digital Medicine, 2(31), 2019.

Josh Barnes and Piet Hut. A hierarchical $\mathcal{O}(N \log N)$ force-calculation algorithm. Nature, 324 (6096):446–449, 1986.

Alina Jade Barnett, Fides Regina Schwartz, Chaofan Tao, Chaofan Chen, Yinhao Ren, Joseph Y. Lo, and Cynthia Rudin. IAIA-BL: A case-based interpretable deep learning model for classification of mass lesions in digital mammography. CoRR arXiv preprint 2103.12308, abs/2103.12308, 2021. URL https://arxiv.org/abs/2103.12308.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, pages 2499–2509, 2018.

Bruce Guenther Baumgart. Geometric modeling for computer vision. Technical report, Stanford University Department of Computer Science, 1974.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In Proceedings of Neural Information Processing Systems (NeurIPS), volume 14, pages 585–591. MIT Press, 2001.

Anna C. Belkina, Christopher O. Ciccolella, Rina Anno, Richard Halpert, Josef Spidlen, and Jennifer E. Snyder-Cappione. Automated optimized parameters for t-distributed stochastic neighbor embedding improve visualization and analysis of large datasets. Nature Communications, 10(5415), 2019.

Yoshua Bengio. Learning deep architectures for AI. Now Publishers Inc, 2009.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8):1798–1828, 2013.

K. P. Bennett and J. A. Blue. Optimal decision trees. Technical report, R.P.I. Math Report No. 214, Rensselaer Polytechnic Institute, 1996.

James C Bezdek, Siew K Chuah, and David Leep. Generalized k-nearest neighbor rules. Fuzzy Sets and Systems, 18(3):237–256, 1986.

Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. Explainable machine learning in deployment. In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAT* '20), page 648–657. Association for Computing Machinery, 2020. ISBN 9781450369367.

Adrien Bibal, Michael Lognoul, Alexandre de Streel, and Benoît Frénay. Legal requirements on explainability in machine learning. Artificial Intelligence and Law, July 30, 2020.

Jacob Bien and Robert Tibshirani. Prototype Selection for Interpretable Classification. The Annals of Applied Statistics, pages 2403–2424, 2011.

Lieven Billiet, Sabine Van Huffel, and Vanya Van Belle. Interval coded scoring extensions for larger problems. In Proceedings of the IEEE Symposium on Computers and Communications, pages 198–203. IEEE, 2017.

Lieven Billiet, Sabine Van Huffel, and Vanya Van Belle. Interval Coded Scoring: A toolbox for interpretable scoring systems. PeerJ Computer Science, 4:e150, 04 2018.

Harald Binder and Gerhard Tutz. A comparison of methods for the fitting of generalized additive models. Statistics and Computing, 18(1):87–99, 2008.

Jan Niklas Böhm, Philipp Berens, and Dmitry Kobak. A Unifying Perspective on Neighbor Embeddings along the Attraction-Repulsion Spectrum. arXiv e-print arXiv:2007.08902, 2020.

Roger C Bone, Robert A Balk, Frank B Cerra, R Phillip Dellinger, Alan M Fein, William A Knaus, Roland MH Schein, and William J Sibbald. Definitions for sepsis and organ failure and guidelines for the use of innovative therapies in sepsis. Chest, 101(6):1644–1655, 1992.

Olivier Bousquet and André Elisseeff. Stability and generalization. Journal of Machine Learning Research, 2(Mar):499–526, 2002.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth, 1984.

Leo Breiman et al. Statistical modeling: The two cultures (with comments and a rejoinder by the author). Statistical Science, 16(3):199–231, 2001.

Miles Brundage, Shahar Avin, Jasmine Wang, Haydn Belfield, Gretchen Krueger, and Gillian Hadfield et al. Toward trustworthy AI development: Mechanisms for supporting verifiable claims. arXiv e-print arXiv:2004.07213, 2020.

Ernest W Burgess. Factors determining success or failure on parole. The workings of the indeterminate sentence law and the parole system in Illinois, pages 221–234, 1928.

Henrik EC Cao, Riku Sarlin, and Alexander Jung. Learning explainable decision rules via maximum satisfiability. IEEE Access, 8:218180–218185, 2020.

Dallas Card, Michael Zhang, and Noah A Smith. Deep Weighted Averaging Classifiers. In Proceedings of the Conference on Fairness, Accountability, and Transparency, pages 369–378, 2019.

Emilio Carrizosa, Amaya Nogales-Gómez, and Dolores Romero Morales. Strongly agree or strongly disagree?: Rating features in support vector machines. Information Sciences, 329: 256 – 273, 2016. Special issue on Discovery Science.

Emilio Carrizosa, Cristina Molero-Rıo, and Dolores Romero Morales. Mathematical optimization in classification and regression trees. TOP, An Official Journal of the Spanish Society of Statistics and Operations Research, 29:5–33, 2021.

Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pages 1721–1730, 2015.

David M Chan, Roshan Rao, Forrest Huang, and John F Canny. Gpu accelerated t-distributed stochastic neighbor embedding. Journal of Parallel and Distributed Computing, 131:1–13, 2019.

Pete Chapman et al. CRISP-DM 1.0 - step-by-step data mining guide. Technical report, SPSS, 2000.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. arXiv e-print arXiv:1611.01838, 2016.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. Journal of Statistical Mechanics: Theory and Experiment, 2019(12): 124018, 2019.

Chaofan Chen and Cynthia Rudin. An optimization approach to learning falling rule lists. In Proceedings of Artificial Intelligence and Statistics (AISTATS), pages 604–612. PMLR, 2018.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This Looks Like That: Deep Learning for Interpretable Image Recognition. In Proceedings of Neural Information Processing Systems (NeurIPS), pages 8930–8941, 2019.

Chaofan Chen, Kangcheng Lin, Cynthia Rudin, Yaron Shaposhnik, Sijia Wang, and Tong Wang. An interpretable model with globally consistent explanations for credit risk. Decision Support Systems, 2021a. Accepted.

Ricky T. Q. Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, Proceedings of Neural Information Processing Systems (NeurIPS), volume 31, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In Proceedings of Neural Information Processing Systems (NeurIPS), pages 2172–2180, 2016.

Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. Nature Machine Intelligence, 2(12):772–782, Dec 2020.

Zhi Chen, Alex Ogren, Cynthia Rudin, Cate Brinson, and Chiara Daraio. How to see hidden patterns in metamaterials with interpretable machine learning. (in progress), 2021b.

Yann Chevaleyre, Frederic Koriche, and Jean-Daniel Zucker. Rounding methods for discrete linear classification. In Proceedings of International Conference on Machine Learning (ICML), pages 651–659, 2013.

Evangelia Christodoulou, Jie Ma, Gary S. Collins, Ewout W. Steyerberg, Jan Y. Verbakel, and Ben Van Calster. A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. Journal of Clinical Epidemiology, 110:12–22, June 2019.

Peter Clark and Robin Boswell. Rule induction with CN2: Some recent improvements. In European Working Session on Learning, pages 151–163. Springer, 1991.

Peter Clark and Tim Niblett. The CN2 induction algorithm. Machine Learning, 3(4):261–283, 1989.

Andy Coenen and Adam Pearce. Understanding UMAP. https://pair-code.github.io/understanding-umap/, 2019.

William W Cohen. Fast effective rule induction. In Proceedings of the Twelfth International Conference on Machine Learning (ICML), pages 115–123, 1995.

Youri Coppens, Kyriakos Efthymiadis, Tom Lenaerts, Ann Nowé, Tim Miller, Rosina Weber, and Daniele Magazzeni. Distilling deep reinforcement learning policies in soft decision trees. In Proceedings of the IJCAI 2019 Workshop on Explainable Artificial Intelligence, pages 1–6, 2019.

Amanda Coston, Ashesh Rambachan, and Alexandra Chouldechova. Characterizing fairness over the set of good models under selective labels. arXiv e-print arXiv:2101.00352, 2021.

T. Cover and P. Hart. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1):21–27, 1967.

Mark W Craven. Extracting comprehensible models from trained neural networks. PhD thesis, University of Wisconsin-Madison Department of Computer Sciences, 1996.

Mark W. Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In Proceedings of the 8th International Conference on Neural Information Processing Systems (NIPS), page 24–30, 1995.

Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In Proceedings of AAAI Conference on Artificial Intelligence (AAAI), volume 33, pages 3412–3420, 2019.

Şeyda Ertekin and Cynthia Rudin. On equivalence relationships between classification and ranking algorithms. Journal of Machine Learning Research, 12:2905–2929, 2011.

Nathan Dahlin, Krishna Chaitanya Kalagarla, Nikhil Naik, Rahul Jain, and Pierluigi Nuzzo. Designing interpretable approximations to deep reinforcement learning with soft decision trees. arXiv e-print arXiv:2010.14785, 2020.

Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D Hoffman, et al. Underspecification presents challenges for credibility in modern machine learning. arXiv e-print arXiv:2011.03395, 2020.

Srijita Das, Sriraam Natarajan, Kaushik Roy, Ronald Parr, and Kristian Kersting. Fitted Q-learning for relational domains. arXiv e-print arXiv:2006.05595, 2020.

Subhajit Das, Dylan Cashman, Remco Chang, and Alex Endert. BEAMES: Interactive multimodel steering, selection, and inspection for regression tasks. IEEE Computer Graphics and Applications, 39(5):20–32, 2019.

Sanjeeb Dash, Oktay Günlük, and Dennis Wei. Boolean decision rules via column generation. In Proceedings of Neural Information Processing Systems (NeurIPS), volume 31, pages 4655–4665, 2018.

Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J Stuckey. MurTree: Optimal classification trees via dynamic programming and search. arXiv e-print arXiv:2007.12652, 2020.

Guillaume Desjardins, Aaron Courville, and Yoshua Bengio. Disentangling factors of variation via generative entangling. arXiv e-print arXiv:1210.5474, 2012.

Sunny Dhamnani, Dhruv Singal, Ritwik Sinha, Tharun Mohandoss, and Manish Dash. RAPID: Rapid and precise interpretable decision sets. In 2019 IEEE International Conference on Big Data (Big Data), pages 1292–1301. IEEE, 2019.

Yashesh Dhebar, Kalyanmoy Deb, Subramanya Nageshrao, Ling Zhu, and Dimitar Filev. Interpretable-AI policies using evolutionary nonlinear decision trees for discrete action systems. arXiv e-print arXiv:2009.09521, 2020.

Botty Dimanov, Umang Bhatt, Mateja Jamnik, and Adrian Weller. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In 24th European Conference on Artificial Intelligence (ECAI), 2020.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. arXiv e-print arXiv:1703.04933, 2017.

Kien Do and Truyen Tran. Theory and evaluation metrics for learning disentangled representations. arXiv e-print arXiv:1908.09961, 2019.

David Dobkin, Truxton Fulton, Dimitrios Gunopulos, Simon Kasif, and Steven Salzberg. Induction of shallow decision trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997.

Jiayun Dong and Cynthia Rudin. Exploring the cloud of variable importance for the set of all good models. Nature Machine Intelligence, 2(12):810–824, 2020.

David L. Donoho and Carrie Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. Proceedings of the National Academy of Arts and Sciences, 100(10): 5591–5596, 2003.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

S. A. Dudani. The Distance-Weighted k-Nearest-Neighbor Rule. IEEE Transactions on Systems, Man, and Cybernetics, SMC-6(4):325–327, 1976.

Sašo Džeroski, Luc De Raedt, and Hendrik Blockeel. Relational reinforcement learning. In International Conference on Inductive Logic Programming, pages 11–22. Springer, 1998.

Sašo Džeroski, Luc De Raedt, and Kurt Driessens. Relational reinforcement learning. Machine Learning, 43(1-2):7–52, 2001.

Cian Eastwood and Christopher KI Williams. A framework for the quantitative evaluation of disentangled representations. In Proceedings of the International Conference on Learning Representations (ICLR), 2018.

Tapio Elomaa. In defense of c4.5: Notes on learning one-level decision trees. In Proceedings of International Conference on Machine Learning (ICML), pages 62–69. Morgan Kaufmann, 1994.

A. Farhangfar, R. Greiner, and M. Zinkevich. A fast way to produce optimal fixed-depth decision trees. In International Symposium on Artificial Intelligence and Mathematics (ISAIM), 2008.

Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. AI Magazine, 17:37–54, 1996.

Sarah Finney, Natalia Gardiol, Leslie Pack Kaelbling, and Tim Oates. The thing that we tried didn't work very well: Deictic representation in reinforcement learning. arXiv e-print arXiv:1301.0567, 2012.

Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. Journal of Machine Learning Research, 20(177):1–81, 2019.

E. Fix and J.L. Hodges. Discriminatory analysis, nonparametric discrimination: Consistency properties. Technical Report 4, 1951.

Eibe Frank and Ian H Witten. Generating accurate rule sets without global optimization. In Proceedings of International Conference on Machine Learning (ICML), pages 144–151, 1998.

Alex A Freitas. Comprehensible classification models: a position paper. ACM SIGKDD Explorations Newsletter, 15(1):1–10, 2014.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119–139, 1997.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. Annals of Statistics, pages 1189–1232, 2001.

Jerome H Friedman. Stochastic gradient boosting. Computational Statistics & Data Analysis, 38 (4):367–378, 2002.

Jerome H Friedman and Nicholas I Fisher. Bump hunting in high-dimensional data. Statistics and Computing, 9(2):123–143, 1999.

Cong Fu, Yonghui Zhang, Deng Cai, and Xiang Ren. Atsne: Efficient and robust visualization on gpu through hierarchical optimization. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), page 176–186, New York, NY, USA, 2019.

Keinosuke Fukunaga and L Hostetler. Optimization of k nearest neighbor density estimates. IEEE Transactions on Information Theory, 19(3):320–326, 1973.

Brian F Gage, Amy D Waterman, William Shannon, Michael Boechler, Michael W Rich, and Martha J Radford. Validation of clinical classification schemes for predicting stroke: results from the national registry of atrial fibrillation. Journal of the American Medical Association (JAMA), 285(22):2864–2870, 2001.

Brian R. Gaines and Paul Compton. Induction of ripple-down rules applied to modeling large databases. Journal of Intelligent Information Systems, 5(3):211–228, 1995.

Bishwamittra Ghosh and Kuldeep S Meel. IMLI: An incremental framework for MaxSAT-based learning of interpretable classification rules. In Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society (AIES), pages 203–210, 2019.

Siong Thye Goh and Cynthia Rudin. Box drawings for learning with imbalanced data. In Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), 2014.

Oscar Gomez, Steffen Holter, Jun Yuan, and Enrico Bertini. ViCE: Visual counterfactual explanations for machine learning models. In Proceedings of the 25th International Conference on Intelligent User Interfaces (IUI'20), 2020. URL http://nyuvis-web.poly.edu/projects/fico/intro,https://github.com/nyuvis/ViCE.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Proceedings of Neural Information Processing Systems (NeurIPS), volume 27, pages 2672–2680, 2014.

Alicja Gosiewska and Przemyslaw Biecek. Do not trust additive explanations. arXiv e-print arXiv:1903.11420, 2020.

J Brian Gray and Guangzhe Fan. Classification tree analysis using target. Computational Statistics & Data Analysis, 52(3):1362–1372, 2008.

Charles C. Gross. Genealogy of the "Grandmother Cell". The Neuroscientist, 8(5):512–518, 2002.

Arthur Guez, Robert D Vincent, Massimo Avoli, and Joelle Pineau. Adaptive treatment of epilepsy via batch-mode reinforcement learning. In Proceedings of AAAI Conference on Artificial Intelligence (AAAI), pages 1671–1678, 2008.

Ryuji Hamamoto, Kruthi Suvarna, Masayoshi Yamada, Kazuma Kobayashi, Norio Shinkai, Mototaka Miyake, Masamichi Takahashi, Shunichi Jinnai, Ryo Shimoyama, Akira Sakai, Ken Takasawa, Amina Bolatkan, Kanto Shozu, Ai Dozen, Hidenori Machino, Satoshi Takahashi, Ken Asada, Masaaki Komatsu, Jun Sese, and Syuzo Kaneko. Application of artificial intelligence technology in oncology: Towards the establishment of precision medicine. Cancers, 12(12), 2020.

K. Hammond. Proceedings: Second Case-Based Reasoning Workshop (DARPA). Morgan Kanfmann Publishers, Inc., San Mateo, CA, 1989.

Asad Haris, Noah Simon, and Ali Shojaie. Generalized sparse additive models. arXiv e-print arXiv:1903.04641, 2019.

Trevor J Hastie and Robert J Tibshirani. Generalized additive models, volume 43. CRC press, 1990.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In Proceedings of the International Conference on Learning Representations (ICLR), 2017.

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, and Alexander Lerchner. Towards a definition of disentangled representations. arXiv e-print arXiv:1812.02230, 2018.

Geoffrey Hinton. How to represent part-whole hierarchies in a neural network. arXiv e-print arXiv:2102.12627, 2021.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In International Conference on Artificial Neural Networks, pages 44–51. Springer, 2011.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. Neural Computation, 9(1):1–42, 1997.

Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. Machine Learning, 11(1):63–91, 1993.

Dat Hong, Stephen S Baek, and Tong Wang. Interpretable sequence classification via prototype trajectory. arXiv e-print arXiv:2007.01777, 2020.

Hao Hu, Mohamed Siala, Emmanuel Hébrard, and Marie-José Huguet. Learning optimal decision trees with MaxSAT and its integration in Adaboost. In IJCAI-PRICAI 2020, 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence, 2020.

Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. In Proceedings of Neural Information Processing Systems (NeurIPS), 2019.

Alexey Ignatiev, Filipe Pereira, Nina Narodytska, and Joao Marques-Silva. A SAT-based approach to learn explainable decision sets. In International Joint Conference on Automated Reasoning, pages 627–645, 2018.

Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. PLoS One, 9(6):1–12, 06 2014.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. arXiv e-print arXiv:1506.02025, 2015.

Mahsa Oroojeni Mohammad Javad, Stephen Olusegun Agboola, Kamal Jethwani, Abe Zeid, and Sagar Kamarthi. A reinforcement learning–based method for management of type 1 diabetes: exploratory study. JMIR diabetes, 4(3):e12905, 2019.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

Zoe Juozapaitis, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. Explainable reinforcement learning via reward decomposition. In IJCAI/ECAI Workshop on Explainable Artificial Intelligence, 2019.

Rishabh Kabra, Chris Burgess, Loic Matthey, Raphael Lopez Kaufman, Klaus Greff, Malcolm Reynolds, and Alexander Lerchner. Multi-object datasets. https://github.com/deepmind/multi-object-datasets/, 2019.

Sham M Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In Proceedings of Neural Information Processing Systems (NeurIPS), pages 793–800, 2009.

Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, and Jennifer Wortman Vaughan. Interpreting interpretability: Understanding data scientists' use of interpretability tools for machine learning. In Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pages 1–14, 2020.

James M Keller, Michael R Gray, and James A Givens. A fuzzy k-nearest neighbor algorithm. IEEE Transactions on Systems, Man, and Cybernetics, 4:580–585, 1985.

Kristian Kersting and Kurt Driessens. Non-parametric policy gradients: A unified treatment of propositional and relational domains. In Proceedings of the 25th International Conference on Machine Learning, pages 456–463, 2008.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. arXiv e-print arXiv:1609.04836, 2016.

Been Kim, Cynthia Rudin, and Julie A Shah. The Bayesian Case Model: A Generative Approach for Case-Based Reasoning and Prototype Classification. In Proceedings of Neural Information Processing Systems (NeurIPS), volume 27, pages 1952–1960, 2014.

Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, Proceedings of Neural Information Processing Systems (NeurIPS), volume 29, pages 2280–2288, 2016.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In Proceedings of International Conference on Machine Learning (ICML), pages 2668–2677. PMLR, 2018.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. arXiv e-print arXiv:1802.05983, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In 2nd International Conference on Learning Representations (ICLR), 2013.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems, pages 1–18, 2021.

Nicholas Kissel and Lucas Mentch. Forward stability and model path selection. arXiv e-print arXiv:2103.03462, 2021.

William A Knaus, Jack E Zimmerman, Douglas P Wagner, Elizabeth A Draper, and Diane E Lawrence. Apache-acute physiology and chronic health evaluation: a physiologically based classification system. Critical Care Medicine, 9(8):591–597, 1981.

William A Knaus, Elizabeth A Draper, Douglas P Wagner, and Jack E Zimmerman. APACHE II: a severity of disease classification system. Critical Care Medicine, 13(10):818–829, 1985.

William A Knaus, Douglas P Wagner, Elizabeth A Draper, Jack E Zimmerman, Marilyn Bergner, Paulo G Bastos, Carl A Sirio, Donald J Murphy, Ted Lotring, Anne Damiano, et al. The APACHE III prognostic system: risk prediction of hospital mortality for critically ill hospitalized adults. Chest, 100(6):1619–1636, 1991.

Dmitry Kobak and Philipp Berens. The art of using t-SNE for single-cell transcriptomics. Nature Communication, 10:5416, 2019.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. The International Journal of Robotics Research, 32(11):1238–1274, 2013.

Yves Kodratoff. The comprehensibility manifesto. KDD Nugget Newsletter, 94(9), 1994.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In Proceedings of International Conference on Machine Learning (ICML), pages 5338–5348. PMLR, 2020.

Teuvo Kohonen. Learning Vector Quantization. The Handbook of Brain Theory and Neural Networks, page 537–540, 1995.

J. L. Kolodner. Proceedings of the Case-Based Reasoning Workshop (DARPA). Morgan Kaufmmnn Publishers, Inc., San Mateo, CA, 1988.

Vladimir Koltchinskii and Dmitry Panchenko. Empirical margin distributions and bounding the generalization error of combined classifiers. The Annals of Statistics, 30(1):1–50, 2002.

Matthieu Komorowski, Leo A Celi, Omar Badawi, Anthony C Gordon, and A Aldo Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. Nature Medicine, 24(11):1716–1720, 2018.

Adam Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. In Proceedings of Neural Information Processing Systems (NeurIPS), pages 15512–15522, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), May 2017.

Tejas D Kulkarni, Will Whitney, Pushmeet Kohli, and Joshua B Tenenbaum. Deep convolutional inverse graphics network. arXiv e-print arXiv:1503.03167, 2015.

Ugur Kursuncu, Manas Gaur, and Amit Sheth. Knowledge Infused Learning (K-IL): Towards Deep Incorporation of Knowledge in Deep Learning. In Proceedings of the AAAI 2020 Spring Symposium on Combining Machine Learning and Knowledge Engineering in Practice (AAAI-MAKE 2020), volume I, 2020.

I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou. Neural-network methods for boundary value problems with irregular boundaries. IEEE Transactions on Neural Networks, 11(5):1041–1049, 2000.

Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks, 9(5):987–1000, 1998.

Himabindu Lakkaraju and Osbert Bastani. "How do I fool you?": Manipulating user trust via misleading black box explanations. In Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES), page 79–85, February 2020.

Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pages 1675–1684, 2016.

Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking Clever Hans predictors and assessing what machines really learn. Nature Communications, 10(1096), 2019.

Thibault Laugel, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pages 2801–2807, 7 2019.

Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is NP-complete. Information Processing Letters, 5(1):15–17, 1976.

Jean-Roger Le Gall, Stanley Lemeshow, and Fabienne Saulnier. A new simplified acute physiology score (SAPS II) based on a European/North American multicenter study. Journal of the American Medical Association (JAMA), 270(24):2957–2963, 1993.

Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. ATT Labs [Online]. Available: http://yann. lecun. com/exdb/mnist, 2, 2010.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

Hyuk Lee and In Seok Kang. Neural algorithm for solving differential equations. Journal of Computational Physics, 91(1):110–131, 1990.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Proceedings of International Conference on Machine Learning (ICML), pages 3744–3753. PMLR, 2019.

Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. The Annals of Applied Statistics, 9(3):1350–1371, 2015.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. arXiv e-print arXiv:1712.09913, 2017.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. arXiv e-print arXiv:1606.01541, 2016.

Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep Learning for Case-Based Reasoning through Prototypes: A Neural Network that Explains Its Predictions. In Proceedings of AAAI Conference on Artificial Intelligence (AAAI), 2018.

Wenmin Li, Jiawei Han, and Jian Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In Proceedings 2001 IEEE International Conference on Data Mining (ICDM), pages 369–376. IEEE, 2001.

Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In Proceedings of International Conference on Machine Learning (ICML), pages 6150–6160. PMLR, 2020.

Yi Lin, Hao Helen Zhang, et al. Component selection and smoothing in multivariate nonparametric regression. The Annals of Statistics, 34(5):2272–2297, 2006.

George C. Linderman, Manas Rachh, Jeremy G. Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. Nature Methods, 16:243–245, 2019.

Bing Liu, Wynne Hsu, Yiming Ma, et al. Integrating classification and association rule mining. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), volume 98, pages 80–86, 1998.

Guiliang Liu, Oliver Schulte, Wang Zhu, and Qingcan Li. Toward interpretable deep reinforcement learning with linear model u-trees. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pages 414–429. Springer, 2018.

Ying Liu, Brent Logan, Ning Liu, Zhiyuan Xu, Jian Tang, and Yangzhi Wang. Deep reinforcement learning for dynamic treatment regimes on medical registry data. In 2017 IEEE International Conference on Healthcare Informatics (ICHI), pages 380–385. IEEE, 2017.

Samuele Lo Piano. Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward. Humanit Soc Sci Commun, 7(9), 2020.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. arXiv e-print arXiv:2006.15055, 2020.

Wei-Yin Loh and Yu-Shan Shih. Split selection methods for classification trees. Statistica Sinica, 7:815–840, 1997.

Max Losch, Mario Fritz, and Bernt Schiele. Interpretability beyond classification output: Semantic bottleneck networks. arXiv e-print arXiv:1907.10882, 2019.

Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pages 150–158, 2012.

Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pages 623–631, 2013.

Yin Lou, Jacob Bien, Rich Caruana, and Johannes Gehrke. Sparse partially linear additive models. Journal of Computational and Graphical Statistics, 25(4):1126–1140, 2016.

Dmitry Malioutov and Kuldeep S Meel. MLIC: A MaxSAT-based framework for learning interpretable classification rules. In International Conference on Principles and Practice of Constraint Programming, pages 312–327. Springer, 2018.

Dmitry Malioutov and Kush Varshney. Exact rule learning via boolean compressed sensing. In Proceedings of International Conference on Machine Learning (ICML), pages 765–773. PMLR, 2013.

Mario Marchand and John Shawe-Taylor. The set covering machine. Journal of Machine Learning Research, 3(Dec):723–746, 2002.

Mario Marchand and Marina Sokolova. Learning with decision lists of data-dependent features. Journal of Machine Learning Research, 6(Apr):427–451, 2005.

Charles Marx, Flavio Calmon, and Berk Ustun. Predictive multiplicity in classification. In Proceedings of International Conference on Machine Learning (ICML), pages 6765–6774. PMLR, 2020.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Michael McGough. How bad is sacramento's air, exactly? Google results appear at odds with reality, some say. Sacramento Bee, August 7 2018.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv e-print arXiv:1802.03426, February 2018.

Manish Mehta, Rakesh Agrawal, and Jorma Rissanen. SLIQ: A fast scalable classifier for data mining. In International Conference on Extending Database Technology, pages 18–32. Springer, 1996.

Lukas Meier, Sara Van de Geer, Peter Bühlmann, et al. High-dimensional additive modeling. The Annals of Statistics, 37(6B):3779–3821, 2009.

Xuhui Meng, Zhen Li, Dongkun Zhang, and George Em Karniadakis. PPINN: Parareal physics-informed neural network for time-dependent pdes. Computer Methods in Applied Mechanics and Engineering, 370:113250, 2020.

Matt Menickelly, Oktay Günlük, Jayant Kalagnanam, and Katya Scheinberg. Optimal decision trees for categorical data via integer programming. arXiv e-print arXiv:1612.03225, January 2018.

George Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. The Psychological Review, 63:81–97, 1956.

Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. Interpretable and Steerable Sequence Learning via Prototypes. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pages 903–913, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. arXiv e-print arXiv:1312.5602, 2013.

Rui P Moreno, Philipp GH Metnitz, Eduardo Almeida, Barbara Jordan, Peter Bauer, Ricardo Abizanda Campos, Gaetano Iapichino, David Edbrooke, Maurizia Capuzzo, Jean-Roger Le Gall, et al. SAPS 3 — from evaluation of the patient to evaluation of the intensive care unit. Part 2: Development of a prognostic model for hospital mortality at ICU admission. Intensive Care Medicine, 31(10):1345–1355, 2005.

James N. Morgan and John A. Sonquist. Problems in the analysis of survey data, and a proposal. J. Amer. Statist. Assoc., 58:415–434, 1963.

Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende. Towards interpretable reinforcement learning using attention augmented agents. In Proceedings of Neural Information Processing Systems (NeurIPS), pages 12350–12359, 2019.

Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and Joao Marques-Silva. Learning optimal decision trees with SAT. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI), pages 1362–1368, 2018.

Meike Nauta, Annemarie Jutte, Jesper Provoost, and Christin Seifert. This Looks Like That, Because... Explaining Prototypes for Interpretable Image Recognition. arXiv e-print arXiv:2011.02863, 2020a.

Meike Nauta, Ron van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. arXiv e-print arXiv:2012.02046, 2020b.

John Ashworth Nelder and Robert WM Wedderburn. Generalized linear models. Journal of the Royal Statistical Society: Series A (General), 135(3):370–384, 1972.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In NIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.

Allen Newell, Herbert Alexander Simon, et al. Human Problem Solving. Prentice-Hall Englewood Cliffs, NJ, 1972.

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

Siegfried Nijssen and Elisa Fromont. Mining optimal decision trees from itemset lattices. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pages 530–539. ACM, 2007.

Siegfried Nijssen, Pierre Schaus, et al. Learning optimal decision trees using caching branch-and-bound search. In Proceedings of AAAI Conference on Artificial Intelligence (AAAI), 2020.

Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability. arXiv e-print arXiv:1909.09223, 2019.

Northpointe. Practitioner's Guide to COMPAS Core. Technical report, Northpointe, March 2013.

Matt O'Connor. Algorithm's 'unexpected' weakness raises larger concerns about AI's potential in broader populations. Health Imaging, Artificial Intelligence section, April 05, 2021.

Guofei Pang, Lu Lu, and George Em Karniadakis. fPINNs: Fractional physics-informed neural networks. SIAM Journal on Scientific Computing, 41(4):A2603–A2626, 2019a.

Guofei Pang, Liu Yang, and George Em Karniadakis. Neural-net-induced gaussian process regression for function approximation and pde solution. Journal of Computational Physics, 384: 270–288, 2019b.

Athanassios Papagelis and Dimitrios Kalles. GA tree: genetically evolved decision trees. In Proceedings 12th IEEE Internationals Conference on Tools with Artificial Intelligence (ICTAI), pages 203–206. IEEE, 2000.

Nicolas Papernot and Patrick McDaniel. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. arXiv e-print arXiv:1803.04765, 2018.

Sonali Parbhoo, Jasmina Bogojeska, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Combining kernel and model based learning for HIV therapy selection. AMIA Summits on Translational Science Proceedings, 2017:239, 2017.

Harsh Parikh, Cynthia Rudin, and Alexander Volfovsky. An application of matching after learning to stretch (MALTS) to the ACIC 2018 causal inference challenge data. Observational Studies, pages 118–130, 2019.

Karl Pearson. On lines and planes of closest fit to systems of points in space. Philosophical Magazine, 2(11):559–572, 1901.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

Ashley Petersen, Daniela Witten, and Noah Simon. Fused lasso additive model. Journal of Computational and Graphical Statistics, 25(4):1005–1025, 2016.

Pavlin G. Poličar, Martin Stražar, and Blaž Zupan. openTSNE: a modular python library for t-SNE dimensionality reduction and embedding. bioRxiv, 2019.

Forough Poursabzi-Sangdeh, Daniel G. Goldstein, Jake M. Hofman, Jennifer Wortman Vaughan, and Hanna M. Wallach. Manipulating and measuring model interpretability. arXiv e-print arXiv:1802.07810, 2018.

Dimitris C Psichogios and Lyle H Ungar. A hybrid neural network-first principles approach to process modeling. AIChE Journal, 38(10):1499–1511, 1992.

J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. Journal of Machine Learning Research, 19(25):1–24, 2018.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems. arXiv e-print arXiv:1801.01236, 2018.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics, 378:686–707, 2019.

Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science, 367(6481):1026–1030, 2020.

Chengping Rao, Hao Sun, and Yang Liu. Physics informed deep learning for computational elastodynamics without labeled data. arXiv e-print arXiv:2006.08472, 2020.

Pradeep Ravikumar, John Lafferty, Han Liu, and Larry Wasserman. Sparse additive models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 71(5):1009–1030, 2009.

C. K. Riesbeck and R. S. Schank. Inside Case-Based Reasoning. Lawrence Erlbaum Assoc., Inc., Hillsdale, NJ, 1989.

Ronald L Rivest. Learning decision lists. Machine learning, 2(3):229–246, 1987.

Aaron M Roth, Nicholay Topin, Pooyan Jamshidi, and Manuela Veloso. Conservative Q-improvement: Reinforcement learning for an interpretable decision-tree policy. arXiv e-print arXiv:1907.01180, 2019.

Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290(5500):2323–2326, 2000.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence, 1:206–215, May 2019.

Cynthia Rudin and Seyda Ertekin. Learning customized and optimized lists of rules with mathematical programming. Mathematical Programming C (Computation), 10:659–702, 2018.

Cynthia Rudin and Joanna Radin. Why are we using black box models in AI when we don't need to? A lesson from an explainable AI competition. Harvard Data Science Review, 1(2), 2019.

Cynthia Rudin and Robert E. Schapire. Margin-based ranking and an equivalence between AdaBoost and RankBoost. Journal of Machine Learning Research, 10:2193–2232, Oct 2009.

Cynthia Rudin and Yaron Shaposhnik. Globally-consistent rule-based summary-explanations for machine learning models: Application to credit-risk evaluation. May 28, 2019. Available at SSRN: https://ssrn.com/abstract=3395422, 2019.

Cynthia Rudin and Berk Ustun. Optimized scoring systems: Toward trust in machine learning for healthcare and criminal justice. Interfaces, 48(5):449–466, 2018.

Cynthia Rudin and Kiri L. Wagstaff. Machine learning for science and society. Machine Learning, 95(1), 2014.

Cynthia Rudin, Benjamin Letham, and David Madigan. Learning theory analysis for association rules and sequential event prediction. Journal of Machine Learning Research, 14(1):3441–3492, 2013.

Cynthia Rudin, Caroline Wang, and Beau Coker. The age of secrecy and unfairness in recidivism prediction. Harvard Data Science Review, 2(1), 2020.

Dawid Rymarczyk, Łukasz Struski, Jacek Tabor, and Bartosz Zieliński. ProtoPShare: Prototype Sharing for Interpretable Image Classification and Similarity Discovery. arXiv e-print arXiv:2011.14340, 2020.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In Proceedings of Neural Information Processing Systems (NeurIPS), pages 3856–3866, 2017.

Veeranjaneyulu Sadhanala, Ryan J Tibshirani, et al. Additive models with trend filtering. The Annals of Statistics, 47(6):3032–3068, 2019.

Francisco Sahli Costabal, Yibo Yang, Paris Perdikaris, Daniel E Hurtado, and Ellen Kuhl. Physics-informed neural networks for cardiac activation mapping. Frontiers in Physics, 8:42, 2020.

Ruslan Salakhutdinov and Geoff Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In Artificial Intelligence and Statistics, pages 412–419, 2007.

Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. Deep reinforcement learning framework for autonomous driving. Electronic Imaging, 2017(19):70–76, 2017.

Robert E Schapire, Yoav Freund, Peter Bartlett, Wee Sun Lee, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. The Annals of Statistics, 26(5):1651–1686, 1998.

Jürgen Schmidhuber. Learning factorial codes by predictability minimization. Neural computation, 4(6):863–879, 1992.

Patrick Schramowski, Wolfgang Stammer, Stefano Teso, Anna Brugger, Franziska Herbert, Xiaoting Shao, Hans-Georg Luigs, Anne-Katrin Mahlein, and Kristian Kersting. Making deep neural networks right for the right scientific reasons by interacting with their explanations. Nature Machine Intelligence, 2(8):476–486, 2020.

Lesia Semenova, Cynthia Rudin, and Ronald Parr. A study in rashomon curves and volumes: A new perspective on generalization and model simplicity in machine learning. arXiv e-print arXiv:1908.01755, 2019.

Sungyong Seo and Yan Liu. Differentiable Physics-informed Graph Networks. arXiv e-print arXiv:1902.02950, 2019.

Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. arXiv e-print arXiv:1712.07294, 2017.

Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In International Conference on Artificial Intelligence and Statistics, pages 1855–1865. PMLR, 2020.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587):484–489, 2016.

AJ Six, BE Backus, and JC Kelder. Chest pain in the emergency room: value of the heart score. Netherlands Heart Journal, 16(6):191–196, 2008.

Marina Sokolova, Mario Marchand, Nathalie Japkowicz, and John S Shawe-taylor. The decision list machine. In Proceedings of Neural Information Processing Systems (NeurIPS), pages 945–952, 2003.

Nataliya Sokolovska, Yann Chevaleyre, Karine Clément, and Jean-Daniel Zucker. The fused lasso penalty for learning interpretable medical scoring systems. In 2017 International Joint Conference on Neural Networks (IJCNN), pages 4504–4511. IEEE, 2017.

Nataliya Sokolovska, Yann Chevaleyre, and Jean-Daniel Zucker. A provable algorithm for learning interpretable scoring systems. In Proceedings of Machine Learning Research Vol. 84: Artificial Intelligence and Statistics (AISTATS), pages 566–574, 2018.

David Spiegelhalter. Should we trust algorithms? Harvard Data Science Review, 2(1), 2020.

Nathan Srebro, Karthik Sridharan, and Ambuj Tewari. Smoothness, low noise and fast rates. In Proceedings of Neural Information Processing Systems (NeurIPS), pages 2199–2207, 2010.

Aaron F. Struck, Berk Ustun, Andres Rodriguez Ruiz, Jong Woo Lee, Suzette M. LaRoche, Lawrence J. Hirsch, Emily J. Gilmore, Jan Vlachy, Hiba Arif Haider, Cynthia Rudin, and M. Brandon Westover. Association of an electroencephalography-based risk score with seizure probability in hospitalized patients. JAMA Neurology, 74(12):1419–1424, December 2017.

Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

Lei Tai, Jingwei Zhang, Ming Liu, Joschka Boedecker, and Wolfram Burgard. A survey of deep network solutions for learning control in robotics: From reinforcement to imitation. arXiv e-print arXiv:1612.07139, 2016.

Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In Proceedings of the 25th International Conference on the World Wide Web, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. Science, 290(5500):2319–2323, 2000.

Martin Than, Dylan Flaws, Sharon Sanders, Jenny Doust, Paul Glasziou, Jeffery Kline, Sally Aldous, Richard Troughton, Christopher Reid, William A Parsonage, et al. Development and validation of the emergency department assessment of chest pain score and 2h accelerated diagnostic protocol. Emergency Medicine Australasia, 26(1):34–44, 2014.

The Smithsonian Institute. Mammuthus primigenius (blumbach). https://3d.si.edu/object/3d/mammuthus-primigenius-blumbach:341c96cd-f967-4540-8ed1-d3fc56d31f12, 2020.

Scott Thiebes, Sebastian Lins, and Ali Sunyaev. Trustworthy artificial intelligence. Electronic Markets, 2020.

Richard Tomsett, Alun Preece, Dave Braines, Federico Cerutti, Supriyo Chakraborty, Mani Srivastava, Gavin Pearson, and Lance Kaplan. Rapid trust calibration through interpretable and uncertainty-aware AI. Patterns, 1(4):100049, 2020.

Warren Torgerson. Multidimensional scaling: I theory and method. Psychometrika, 17(4):401–419, 1952.

Loc Trinh, Michael Tsang, Sirisha Rambhatla, and Yan Liu. Interpretable and trustworthy deepfake detection via dynamic prototypes. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV, 2021), pages 1973–1983, 2021.

Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. Machine Learning, 102(3):349–391, 2016.

Berk Ustun and Cynthia Rudin. Optimized risk scores. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pages 1125–1134, 2017.

Berk Ustun and Cynthia Rudin. Learning optimized risk scores. Journal of Machine Learning Research, 20(150):1–75, 2019.

Berk Ustun, Stefano Traca, and Cynthia Rudin. Supersparse linear integer models for predictive scoring systems. arXiv e-print arXiv:1306.5860, 2013.

Berk Ustun, M.B. Westover, Cynthia Rudin, and Matt T. Bianchi. Clinical prediction models for sleep apnea: The importance of medical history over symptoms. Journal of Clinical Sleep Medicine, 12(2):161–168, 2016.

Laurens van der Maaten. Accelerating t-SNE using tree-based algorithms. Journal of Machine Learning Research, 15:3221–3245, 2014.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. Journal of Machine Learning Research, 9:2579–2605, 2008.

Koen Vanhoof and Benoît Depaire. Structure of association rule classifiers: a review. In 2010 IEEE International Conference on Intelligent Systems and Knowledge Engineering (ISKE), pages 9–12. IEEE, 2010.

VN Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability and its Applications, 16(2):264, 1971.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. arXiv e-print arXiv:1804.02477, 2018.

Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In Proceedings of AAAI Conference on Artificial Intelligence (AAAI), 2019.

Matheus Guedes Vilas Boas, Haroldo Gambini Santos, Luiz Henrique de Campos Merschmann, and Greet Vanden Berghe. Optimal decision trees for the algorithm selection problem: integer programming based approaches. International Transactions in Operational Research, 2019.

Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. International Data Privacy Law, 12 2017.

Kiri L. Wagstaff. Machine learning that matters. In Proceedings of International Conference on Machine Learning (ICML), pages 529–536, 2012.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Fulton Wang and Cynthia Rudin. Falling rule lists. In Proceedings of Artificial Intelligence and Statistics (AISTATS), pages 1013–1022, 2015a.

Jiaxuan Wang, Jeeheh Oh, Haozhu Wang, and Jenna Wiens. Learning credible models. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2018.

Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. arXiv e-print arXiv:2001.04536, 2020a.

Tianyu Wang, Marco Morucci, M. Usaid Awan, Yameng Liu, Sudeepa Roy, Cynthia Rudin, and Alexander Volfovsky. Flame: A fast large-scale almost matching exactly approach to causal inference. Journal of Machine Learning Research, 22(31):1–41, 2021a.

Tong Wang. Gaining free or low-cost interpretability with interpretable partial substitute. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 6505–6514. PMLR, 09–15 Jun 2019.

Tong Wang and Cynthia Rudin. Learning optimized Or's of And's. arXiv e-print arXiv:1511.02210, 2015b.

Tong Wang, Jingyi Yang, Yunyi Li, and Boxiang Wang. Partially interpretable estimators (PIE): black-box-refined interpretable machine learning. CoRR arXiv 2105.02410, abs/2105.02410, 2021b. URL https://arxiv.org/abs/2105.02410.

Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: An empirical approach to deciphering t-SNE, UMAP, TriMap, and PaCMAP for data visualization, 2020b.

Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-SNE effectively. Distill, 1 (10):e2, 2016.

Frank W Weathers, Brett T Litz, Terence M Keane, Patrick A Palmieri, Brian P Marx, and Paula P Schnurr. The PTSD checklist for DSM-5 (PCL-5) – standard [measurement instrument]. Scale available from the National Center for PTSD at www. ptsd. va. gov, 10, 2013.

Kilian Q Weinberger and Lawrence K Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. Journal of Machine Learning Research, 10(2), 2009.

Rebecca Wexler. When a computer program keeps you in jail: How computers are harming criminal justice. New York Times, June 13 2017.

Simon N Wood. Generalized additive models: an introduction with R. CRC press, 2017.

Chenyue Wu and Esteban G Tabak. Prototypal Analysis and Prototypal Regression. arXiv e-print arXiv:1701.08916, 2017.

Feng Xie, Bibhas Chakraborty, Marcus Eng Hock Ong, Benjamin Alan Goldstein, and Nan Liu. Autoscore: A machine learning–based automatic clinical score generator and its application to mortality prediction using electronic health records. JMIR Med Inform, 8(10):e21798, October 2020.

Hongyu Yang, Cynthia Rudin, and Margo Seltzer. Scalable bayesian rule lists. In Proceedings of International Conference on Machine Learning (ICML), pages 3921–3930. PMLR, 2017.

Yibo Yang and Paris Perdikaris. Conditional deep surrogate models for stochastic, high-dimensional, and multi-fidelity systems. Computational Mechanics, 64(2):417–434, 2019.

Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. arXiv preprint arXiv:1806.06988, 2018.

Junming Yin and Yaoliang Yu. Convex-constrained sparse additive modeling and its extensions. arXiv e-print arXiv:1705.00687, 2017.

Xiaoxin Yin and Jiawei Han. Cpar: Classification based on predictive association rules. In Proceedings of the 2003 SIAM International Conference on Data Mining (SDM), pages 331–335. SIAM, 2003.

Chao Yu, Jiming Liu, and Shamim Nemati. Reinforcement learning in healthcare: A survey. arXiv e-print arXiv:1908.08796, 2019.

Jinqiang Yu, Alexey Ignatiev, Pierre Le Bodic, and Peter J Stuckey. Optimal decision lists using sat. arXiv e-print arXiv:2010.09919, 2020a.

Jinqiang Yu, Alexey Ignatiev, Peter J Stuckey, and Pierre Le Bodic. Computing optimal decision sets with sat. In International Conference on Principles and Practice of Constraint Programming, pages 952–970. Springer, 2020b.

Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding DQNs. In International Conference on Machine Learning, pages 1899–1908. PMLR, 2016.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. arXiv e-print arXiv:1806.01830, 2018.

John R. Zech et al. Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study. PLoS Med., 15(e1002683), 2018.

Kristine Zhang, Yuanheng Wang, Jianzhun Du, Brian Chu, Leo Anthony Celi, Ryan Kindle, and Finale Doshi-Velez. Identifying decision points for safe and interpretable reinforcement learning in hypotension treatment. arXiv e-print arXiv:2101.03309, 2021.

Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 8827–8836, 2018.

Yujia Zhang, Kuangyan Song, Yiming Sun, Sarah Tan, and Madeleine Udell. "Why should you trust my explanation?" Understanding uncertainty in LIME explanations. In Proceedings of the International Conference on Machine Learning AI for Social Good Workshop, 2019.

Hongxu Zhao, Ran Jin, Su Wu, and Jianjun Shi. PDE-constrained gaussian process model on material removal rate of wire saw slicing process. Journal of Manufacturing Science and Engineering, 133(2), 2011.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene CNNs. In Proceedings of the International Conference on Learning Representations (ICLR), 2014.

Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018a.

Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Interpretable basis decomposition for visual explanation. In Proceedings of the European Conference on Computer Vision (ECCV), pages 119–134, 2018b.

Ding-Xuan Zhou. The covering number in learning theory. Journal of Complexity, 18(3):739–767, 2002.

Yinhao Zhu, Nicholas Zabaras, Phaedon-Stelios Koutsourelakis, and Paris Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. Journal of Computational Physics, 394:56–81, 2019.

Zhenyao Zhu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Multi-view perceptron: a deep model for learning face identity and view representations. In Proceedings of Neural Information Processing Systems (NeurIPS), pages 217–225, 2014.