Category Theory in Machine Learning

Dan Shiebler
University of Oxford
daniel.shiebler@kellog.ox.ac.uk

Bruno Gavranović
University of Strathclyde
bruno@brunogavranovic.com

Paul Wilson
University of Southampton
paul@statusfailed.com

Over the past two decades machine learning has permeated almost every realm of technology. At the same time, many researchers have begun using category theory as a unifying language, facilitating communication between different scientific disciplines. It is therefore unsurprising that there is a burgeoning interest in applying category theory to machine learning. We aim to document the motivations, goals and common themes across these applications. We touch on gradient-based learning, probability, and equivariant learning.

Contents

1 Introduction

Compared to mathematics or physics, machine learning is a young field. Despite its young age, it has experienced sprawling growth, with its applications now permeating almost every realm of technology. A dozen of its subfields have become entire areas of study in their own right. This includes computational learning theory, deep learning, Bayesian inference, normalizing flows, clustering, reinforcement learning, and meta learning.

And yet, this explosive growth has not come without its costs. As the field keeps growing, it is becoming harder and harder to manage its complexity, and to understand how parts of this immense body of research interact with each other. While different subfields of machine learning share the same intellectual framework, it is hard to talk across boundaries. Many subfields have their own theoretical underpinnings, best practices, evaluation measures, and often very different languages and ways of thinking about the field as a whole. Furthermore, the fast-moving pace of the field is giving rise to bad incentives (Britz, 2020), leading to papers with confounding variables, missing details, bad notation, and suffering from narrative fallacy and *research debt* (Olah and Carter, 2017). While these issues are in general not exclusive to machine learning, the subfield of deep learning is notoriously ad-hoc (Olah, 2015). In his NeurIPS Test of Time award speech (Rahimi, 2018), Ali Rahimi has compared modern machine learning to alchemy[1], citing examples of machine learning models performance dropping drastically after

[1] Without our knowledge of modern chemistry chemistry and physics, alchemists attempted to find an elixir for immortality, and cure any disease.

internals of frameworks they used changed the default way of rounding numbers. Models in deep reinforcement learning are particularly brittle, often changing their performance drastically with different initial seeds (Irpan, 2018). In general, the construction of most non-trivial machine learning systems is largely guided by heuristics about what works well in practice. While the individual components of complex models are generally well-developed mathematically, their composition and combinations tend to be poorly understood.

The machine learning community is well aware of this problem. Some researchers have begun to organize workshops focused on the compositionality of machine learning components (Com, 2019). Others have called for broad overarching frameworks to unify machine learning theory and practice (Rieck, 2020). Nonetheless, there does not seem to be widespread consensus about how exactly to achieve that goal.

On the other hand, the field of category theory has steadily been growing. It is becoming a unifying force in mathematics and physics, spreading in recent years into chemistry, statistics, game theory, causality, and database theory. As the science of compositionality, it helps structure thoughts and ideas, find commonalities between different branches of science, and transfer ideas from one field to another (Fong and Spivak, 2018; Bradley, 2018).

Since many modern machine learning systems are inherently compositional (Abadi et al., 2015), this makes it unsurprising that a number of authors have begun to study them through the lens of category theory. In this survey we will describe category theoretic perspectives on three areas:

- **Gradient-based methods.** Building from the foundations of automatic differentiation to neural network architectures, loss functions and model updates.
- **Probabilistic methods.** Building from the foundations of probability to simple Bayesian models.
- **Invariant and Equivariant Learning.** Characterizing the invariances and equivariances of unsupervised and supervised learning algorithms.

While our aim is to provide a comprehensive account of the approaches above, we note that there are many category theoretic perspectives on machine learning that we do not touch on. For example, we leave the field of natural language processing (which has seen recent interest from the categorical community (Brucker, 2020)) to future work.

Notation. In this paper, we write function composition in diagrammatic order using $\fatsemi$. When it comes to the direction of string diagrams, we follow the notation conventions of the authors. Therefore, in the first section, the string diagrams flow from left to right, while in the second section they flow from bottom to top.

2 Gradient-based Learning

2.1 Overview

The research in this section corresponds mostly to deep learning, starting from the foundations of back-propagation – **automatic differentiation** – and moving on to the concept of a neural networks, gradient descent, and a loss function, where updating proceeds in an iterative fashion.

2.1.1 Applications, Successes, and Motivation

Models based on deep neural networks have enjoyed the most high-profile successes of the three fields we discuss. For example, in Reinforcement Learning, AlphaGo (Silver et al., 2017) achieved super-human

performance in playing the game of Go, while OpenAI's GPT-3 (Brown et al., 2020) natural language model is able to generate realistic human-like text. Typical examples of machine learning problems addressable with the methods in this section are:

- **Classification and regression**: Given a dataset of input/output examples (A, B), learning a function $f : A \to B$ mapping inputs to outputs. Classification is when B is a finite set, regression when B is real-valued.
- **Generative models**: Given a dataset of examples, learning to generate new samples which are "close" to those in the dataset. For example, training on and generating images of faces.
- **Reinforcement learning**: Problems framed as an 'agent' taking actions in some 'environment'. For example, the simple 'cart-pole' system, where a cart (the agent) must move along a track in order to balance a pole vertically.

2.1.2 Background

For the purposes of this section, we will take the view that a machine learning model is simply a morphism $f : P \otimes A \to B$ in some monoidal category $(\mathbf{C}, \otimes, I)$, with P an object representing the type of parameters, A representing some observed data, and B some kind of prediction. For example, if our goal is to classify 28×28-pixel images into two classes, we might have [2] $A = \mathbb{R}^{28\times 28}$ and $B = [0, 1]$, with the latter representing a probability.

Training such a model consists of finding a specific parameter value $\theta : I \to P$, thereby giving trained model $f\theta : A \to B$. String-diagramatically:

untrained model trained model

In essence, the parameters P serve to index a collection of maps, and so searching for a map $(\theta \otimes \mathsf{id}) \fatsemi f : A \to B$ reduces to searching for a value $\theta : I \to P$.

2.1.3 Big Ideas and Challenges

We have organised the research in this section into three areas:

- **Computing the Gradient**: Gradient-based optimization is ubiquitous in machine learning–especially neural networks–so computing the gradient efficiently is key. In this section, we discuss categorical approaches to this computation.
- **Learning with Lenses**: Lenses are a specific type of optics which provide read and write access to a value in context. In this section, we explore how lenses can capture the forward 'predictive' and backward 'update' behaviours of learning.
- **Parameter Updates and Learning**: Finally, we discuss how lens-based formalisms for learning capture the various machine learning algorithms used in practice.

[2] Since pixels are not actually real-valued, we may instead use $A = F^{28\times 28}$ where F is the set of all floating point numbers.

Some challenges remain, however. For example, current categorifications of forward and reverse derivatives are 'simply-typed': it is assumed that the type of changes is the same as the type of values. Addressing this would allow for gradient-based learning in categories with more complex structure. Further, there has been little work from a category-theoretic perspective on convergence properties of the procedures discussed here: this may be important for a full end-to-end understanding of learning.

2.2 Computing the Gradient

Gradient descent is a ubiquitous approach for training machine learning models where one views learning a model $f : P \otimes A \to B$ as iteratively improving some initial guess of parameters $\theta : I \to P$ in order to minimise some choice of 'loss' function. The gradient of this loss function is interpreted as the direction of steepest ascent: gradient descent makes repeated steps in the negative direction of the gradient to converge on a local minimum. It is important that the computation of the gradient is efficient, since it must be recomputed at each of the many itererations made by gradient descent.

One of the first incarnations of an algorithm for efficiently computing the gradient of a loss function with respect to some parameters is backprop, originally appearing in the machine learning literature (Rumelhart et al., 1986). The first examination of backpropagation in a categorical setting is the seminal paper "Backprop as functor" (Fong et al., 2019), whose title alone concisely summarises the scope of this section. However, since this paper examines the end-to-end learning process in full, we shall leave its discussion until Section 2.6.

Instead, we begin in Section 2.2.1 with a discussion of Cartesian Differential Categories, which categorify the notion of a differential operator. However, we will see that this is not quite what we need to train a machine learning model via gradient descent: for this, we need a reverse differential operator, which we discuss in Section 2.2.2.

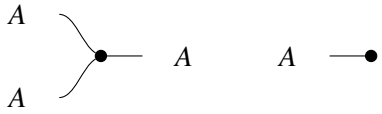
2.2.1 Cartesian Differential Categories

Seely et al. (2009) introduce Cartesian Differential Categories, which are defined as having a **differential combinator** D which sends a map $f : A \to B$ to a generalized derivative map $D[f] : A \times A \to B$. The authors build on their earlier paper (Seely et al., 2006), which defines a **differential category** as additive symmetric monoidal category equipped with a differential combinator and a comonad; **Cartesian differential categories** are introduced to explicitly characterize the notion of an infinitely differentiable category.

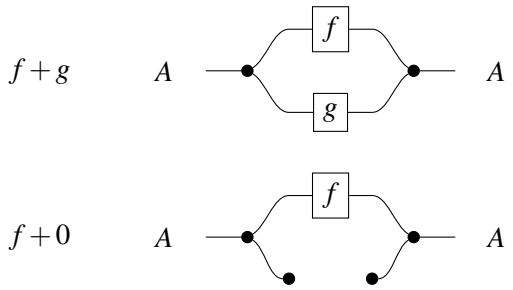
Cartesian Differential Categories are defined in terms of left-additive structure, which we first recall.

Definition 2.1 (Def. 1 in (Cockett et al., 2019))**.** *Let $\mathscr{C}$ be a Cartesian monoidal category. $\mathscr{C}$ is said to be **Cartesian left-additive** when it canonically bears the structure of a commutative monoid with addition $+ : A \times A \to A$ and zero $0_A : 1 \to A$*

Diagrammatically, we write the addition and zero maps as follows

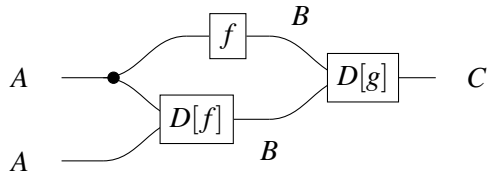


Note that this gives a way to add maps $f, g : \mathscr{C}(A, B)$:



Definition 2.2 (Def. 4 in (Cockett et al., 2019))**.** *A **Cartesian Differential Category** $\mathscr{C}$ is a Cartesian left-additive category equipped with a **differential combinator** D which assigns to each map $f : A \to B$ in $\mathscr{C}$ a map $D[f] : A \times A \to B$, satisfying the equations **CDC.1** to **CDC.7** of (Cockett et al., 2019, Definition 4).*

While we don't list each of the axioms **CDC.1** to **CDC.7**, we highlight one in particular: **CDC.5**. This axiom–the chain rule–defines the differential operation on composite maps.



A familiar example of a reverse differential category is **Smooth**: the category of Euclidean spaces and infinitely differentiable maps between them.

Example 1 (Example 5 in (Cockett et al., 2019))**.** *For a map $f : \mathbb{R}^a \to \mathbb{R}^b$ in* **Smooth***, The derivative $D[f] : \mathbb{R}^a \times \mathbb{R}^a \to \mathbb{R}^b$ is the following smooth function:*

$$D[f](x,x') = J_f(x) \cdot x' = \begin{bmatrix} \frac{\partial df_1}{\partial x'_1} & \cdots & \frac{\partial df_1}{\partial x'_a} \\ \vdots & \ddots & \\ \frac{\partial df_b}{\partial x'_1} & & \frac{\partial df_b}{\partial x'_a} \end{bmatrix} \begin{pmatrix} x'_1 \\ \vdots \\ x'_a \end{pmatrix}$$

Note that $D[f]$ is linear in its first argument, and that it maps a vector of coefficients $x \in \mathbb{R}^a$ and a vector of values $x' \in \mathbb{R}^a$ to the projection of x' along the Jacobian of f at x.

We are to think of the second argument x' as a vector of changes, so that $D[f]$ is the following (linear) approximation:

$$f(x+x') \approx f(x) + D[f](x,x')$$

The particular notion of linearity here is discussed by Seely et al. (2009), where a map f is defined to be linear if $D[f] = \pi_1 f$ (where π_1 is the right projection map). This is a generalization of the idea that the derivative of a linear map is the map itself. Since linear maps are preserved under composition and tensor, any Cartesian differential category contains a subcategory of linear maps.

2.2.2 Reverse Derivative Categories

Although Cartesian differential categories give a suitably generalised definition of the derivative, they do not provide quite what we need for gradient-based learning. Consider the following supervised learning scenario, where we have:

- A parametrised **model** $f : P \times A \to B$
- A **training example** $(a,b) : 1 \to A \times B$
- A choice of **parameters** $\theta : 1 \to P$

Intuitively speaking, we would like to use our training data (a,b) to compute some new parameter $\hat{\theta}$ such that $f(\hat{\theta},a)$ is a better approximation of b than $f(\theta,a)$. The derivative of f gives us a morphism $D[f] : (P \times A) \times (P \times A) \to B$, but what we actually want is a morphism of type $(P \times A) \times B \to P \times A$. This is precisely the type of the reverse derivative combinator introduced by Cockett et al. (2019):

Definition 2.3 (Def. 13 in (Cockett et al., 2019))**.** *A **Reverse Derivative Category** $\mathscr{C}$ is a Cartesian left-additive category equipped with a **reverse derivative combinator** R which assigns to each map $f : A \to B$ in $\mathscr{C}$ a map $R[f] : A \times B \to A$, satisfying the equations **RDC.1** to **RDC.7** of (Cockett et al., 2019, Definition 13).*

We again leave a full listing of the axioms to the original paper (Cockett et al., 2019), but highlight **RDC.5**: the reverse chain rule (Figure 1).

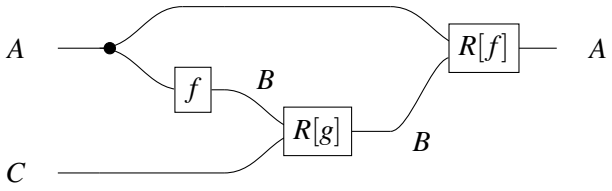


Figure 1: Reverse chain rule in graphical form

Intuitively, this axiom captures the of backpropagation: given a point A, output changes C flow backwards through $R[g]$ and then $R[f]$ to compute a change in the initial inputs, A. Returning to our previous example, **Smooth** also forms a reverse derivative category.

Example 2. *For a map $f : \mathbb{R}^a \to \mathbb{R}^b$ in* **Smooth***, the reverse derivative $R[f] : \mathbb{R}^a \times \mathbb{R}^b \to \mathbb{R}^a$ is the following smooth function:*

$$R[f](x,y') = J_f(x)^T \cdot y'$$

By analogy with the differential combinator D, note that $R[f]$ is also linear in its first argument, and that we are to think of the second argument as a vector of changes, so that $R[f]$ gives the following (linear) approximation:

$$f(x) + y' \approx f(x + R[f](x,y'))$$

2.2.3 Automatic Differentiation

In "The Simple Essence of Automatic Differentiation" (Elliott, 2018), Conal Elliott takes a different perspective. Whereas Fong et al. (2019) first construct a framework for transmitting information through update and request functions and then demonstrate that the BP algorithm fits into that framework , Elliott generalizes the automatic differentiation algorithm itself by replacing linear maps with an arbitrary Cartesian category.

Elliott (2018) writes from a functional programming perspective, and the paper's generalization of automatic differentiation is essentially a program specification. The central construction in the paper is a framework for "differentiable functional programming" that relies on a typed higher-order partial function that tracks and computes derivatives of programs, and Elliott defines efficient implementations of derivatives by using a program transformation strategy (Burstall and Darlington, 1977).

To be specific, Elliott defines the derivative

$$\mathsf{D} : (a \to b) \to (a \to (a \multimap b))$$

that satisfies:

- the **chain rule**: $\mathsf{D}(f \fatsemi g)a = \mathsf{D}fa \fatsemi \mathsf{D}g(fa)$
- the **cross rule**: $\mathsf{D}(f \times g)(a,b) = \mathsf{D}fa \times \mathsf{D}gb$
- the **linear rule**: For all linear functions f, $\mathsf{D}fa = f$

All of these theorems are satisfied by the classical notion of derivatives, and Elliott demonstrates the generality of his construction by relying only on these properties. In order to enable the composition of derivatives, he also defines the operator

$$\mathsf{D}^+ f = (f, \mathsf{D}f)$$

which acts as an endofunctor over a category of differentiable functions.

From the perspective of Elliott's construction, the difference between reverse-mode and forward-mode automatic differentiation reduces to the difference between left/right associated compositions of derivatives. This allows Elliott to define an implementation of reverse-mode differentiation from a continuation-passing style (Kennedy, 2007) rather than dealing with a gradient tape (an additional data structure to store intermediate gradient computations). Furthermore, because the differentiation operator decorates the inference function, Elliott's construction (Elliott, 2018) explicitly specifies how computation between the "forward pass" and "backward pass" are shared in reverse mode automatic differentiation. This is a key aspect of why reverse mode automatic differentiation is particularly efficient for gradient-based learning.

We note that the framework of differential and reverse differential categories introduced in the previous subsection can be seen as a categorical formalization of Elliott's work, if one restricts to the non-closed setting (neither work subsumes the other however, as Elliot's work has a more explicit programming focus). In other words, by taking the codomain $a \to (a \multimap b)$ of Elliott's operator under the tensor-hom adjunction we obtain $a \times a \to b$, which agrees with the differential operator of Cartesian differential categories.

2.3 Optics and Lenses

The story of Cartesian (reverse) differential categories and automatic differentiation lends itself naturally into the story of lenses, and more generally optics. Optics are a general construction which can be

thought of as a pair of processes that move in opposite directions. For example, we might think of the map $f : P \times A \to B$ of a neural network as 'forward' and its reverse derivative $R[f] : P \times A \times B \to P \times A$ as 'backwards'–in the same sense as 'backpropagation'. But more importantly, optics formalize more than just the flow of derivatives and capture a wide array of data accessor patterns (Riley, 2018; Ghani et al., 2016), as well as Bayesian inversion (Section 3.4).

Definition 2.4 (Def. 2.0.1 in (Riley, 2018))**.** *Let $\mathscr{C}$ be a symmetric monoidal category. For any two pairs S, S' and A, A' of objects in $\mathscr{C}$, an* ***optic*** $p : \binom{S'}{S} \to \binom{A'}{A}$ *is an element of the following set:*

$$\int^{M \in C} \mathscr{C}(S, M \otimes A) \times \mathscr{C}(M \otimes A', S')$$

Explicitly, an optic is an equivalence class of triples

$$(M, l : S \to M \otimes A, r : M \otimes A' \to S')$$

such that for any $M_1, M_2 \in C$ and $f : M_1 \to M_2$ in C, we have:

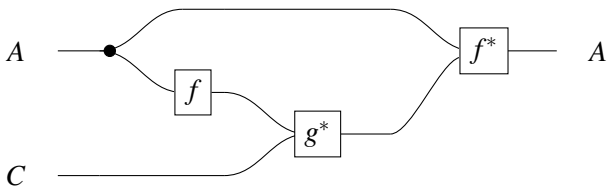
$$(M_1, l \fatsemi (f \otimes id_A), r) \sim (M_2, l, (f \otimes id_A) \fatsemi r)$$

By specializing $\mathscr{C}$ to a Cartesian symmetric monoidal category, we can obtain a concrete description of these optics without the coend (Riley, 2018, Prop. 2.0.4). These constructions are called lenses (Clarke et al., 2020; Bohannon et al., 2008; Román, 2021; Hedges, 2018).

We present a definition of so-called *simple* lenses, whose "forward" and "backward" types are the same.

Definition 2.5 (Def. 1 in (Hedges, 2018))**.** *A* ***Lens*** $A \to B$ *is a pair (f, f^*) of maps $f : A \to B$ and $f^\# : A \times B \to A$*

One can think of lenses as actually having the type $(A, A) \to (B, B)$, where information first flows from the top A to the top B using the map f. The environment then uses the top B to compute and updated B. Together with the original A, this B is used by the map f^* to produce an updated A. In fact, this original A used in the backward pass is exactly the residual when we think of this lens as an optic. In this survey we will write object simply as A, and not as tuples, while keeping in mind that each object actually is used both as an "input" and an "output". Lens composition $(f, f^*) \fatsemi (g, g^*)$ is given by the map $f \fatsemi g$ in the first component and



in the second. We note here the similarity to axiom RDC.5 of reverse differential categories (Figure 1), whose reverse derivatives compose in the same way. This is one of the main insights of (Cruttwell et al., 2021). They show that for any reverse differential category $\mathscr{C}$ there is a canonical embedding into lenses. In other words, each reverse differential function can be augmented with its reverse derivative in a way which respects lens composition.

Proposition 1 (Prop. 2.12 in (Cruttwell et al., 2021))**.** *Let $\mathscr{C}$ be a reverse derivative category. Then there is a canonical, product-preserving, identity-on-objects functor*

$$R : \mathscr{C} \to \mathbf{Lens}(\mathscr{C})$$

which sends a map f to the pair $(f, R[f])$.

This is important since it shows us that reverse derivative categories (Cockett et al., 2019) naturally fit into the story of lenses. This connects two disparate fields of categorical differentiation and bidirectional processes under the common language of lenses.

2.4 Para

While the previous section described categorical foundations behind **backprop**, nothing was said about learning itself. In addition to the forward-backward interaction of derivatives in a machine learning algorithm, behind the scenes there is yet another forward-backward interaction influencing the learning process. This where the story of **parameters** of a machine learning model come in.

Originally described in (Fong et al., 2019), the construction **Para** makes it precise what is meant by parameterization. Here we state the more general version described in (Cruttwell et al., 2021).

Definition 2.6. *Let $\mathscr{C}$ be a symmetric monoidal category. Then* $\mathbf{Para}(\mathscr{C})$ *is a bicategory with the same objects as $\mathscr{C}$. A morphism $A \to B$ in* $\mathbf{Para}(\mathscr{C})$ *is a pair (P, f) where P is an object of $\mathscr{C}$ and*

$$f : P \otimes A \to B$$

A 2-cell from (P, f) to (P', f') is a morphism in $r : P' \to P$ in $\mathscr{C}$ such that the following diagram commutes in $\mathscr{C}$:

$$\begin{array}{ccc} P' \otimes A & \xrightarrow{r \otimes A} & P \otimes A \\ & {\scriptstyle f'} \searrow \quad \swarrow {\scriptstyle f} & \\ & B & \end{array} \tag{1}$$

The composition of 1-cells

$$A \xrightarrow{(P,f)} B \xrightarrow{(Q,g)} C$$

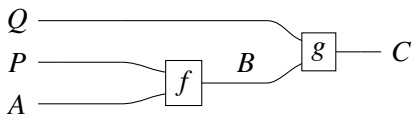
i.e. of

$$P \otimes A \xrightarrow{f} B \qquad \text{and} \qquad Q \otimes B \xrightarrow{g} C$$

is given by $(Q \otimes P, \alpha_{Q,P,A} \fatsemi (Q \otimes f) \fatsemi g)$:

$$(Q \otimes P) \otimes A \xrightarrow{\alpha_{Q,P,A}} Q \otimes (P \otimes A) \xrightarrow{Q \otimes f} Q \otimes B \xrightarrow{g} C$$

Consider a 1-cell $(P, f) : A \to B$ in the bicategory $\mathbf{Para}(\mathscr{C})$. This is a map from A to B with an "extra input" P that has to be accounted for. These inputs can be thought of as the "private knowledge" of the morphism and note that they make sense in any monoidal category $\mathscr{C}$. The composition of two parameterized maps $f : P \otimes A \to B$ and $g : Q \otimes B \to C$ collects the parameters into the monoidal product, as shown in the figure below



The authors show **Para** is also natural with respect to base change. This becomes important when augmenting a differentiable map with its derivative (Prop. 1, and).

Proposition 2 (Prop 2.3 in (Cruttwell et al., 2021))**.** *Let $\mathscr{C}$ and $\mathscr{D}$ be symmetric monoidal categories, and $F : \mathscr{C} \to \mathscr{D}$ a lax symmetric monoidal functor. Then there is an induced lax functor*

$$\mathbf{Para}(F) : \mathbf{Para}(\mathscr{C}) \to \mathbf{Para}(\mathscr{D})$$

The **Para** construction captures the idea that the P inputs of a machine learning model $f : P \otimes A \to B$ are the values to be learned. In the next subsection we describe how the authors in (Cruttwell et al., 2021; Fong et al., 2019) describe these machine learning models.

2.5 Learners

The categorical perspective on neural networks starts with the seminal paper *Backprop as Functor* (Fong et al., 2019). They call machine learning models **learners** and give a concrete description instantiated in the category **Set**. However, this definition is not based on any of the categorical constructions related to differentiation outlined in the previous section. Cruttwell et al. (2021) solve this problem by defining learners as a high-level categorical construct involving **Para**, lenses and reverse derivative categories, subsuming the definition of Fong et al.[3] This is the perspective we take in this text, taking detour to Fong's learners as needed.

Definition 2.7 (Lemma 2.13 in (Cruttwell et al., 2021))**.** *The bicategory of learners is the bicategory* $\mathbf{Para}(\mathbf{Lens}(\mathscr{C}))$*, consisting of the following data:*

- *Objects in* $\mathbf{Para}(\mathbf{Lens}(\mathscr{C}))$ *are the same as those of* $\mathbf{Lens}(\mathscr{C})$*.*
- *A 1-cell $A \to B$ is a bidirectional parameterized lens: a tuple (P, f, f^*) consisting of a choice of a parameter P and a lens $(f, f^*) : P \times A \to B$.*
- *A 2-cell between (P, f, f^*) and $\Rightarrow (Q, g, g^*)$ is a reparameterization lens $(r, r^*) : P \to Q$ satisfying the commuting triangle condition from definition 2.6.*

The 1-cell in this bicategory are the learners, where we think of P the parameters, A as the input to this machine learning model and B as the output. Unpacking this even further, we see that a 1-cell consists of two parts

$$f : P \times A \to B$$
$$f^* : P \times A \times B \to P \times A$$

The map $f : P \times A \to B$ is the map propagates values forward. It takes takes as input a parameter value P, a datapoint A, and computes the "prediction" B. The map f^* performs propagates errors backward: for a value of P, A and B it computes an error for the parameter P and an error for the parameter A, which we think of as the backpropagated error, used by the previous learner. This is the required component to make composition of learners well defined. Originally seen as the necessary, but slightly more mysterious component of (Fong et al., 2019), here it falls out of the definition of lens composition (Cruttwell et al., 2021, Def. 2.7). This is because the composition in $\mathbf{Para}(\mathbf{Lens}(\mathscr{C}))$ is defined in terms of the composition in the base category, which is in this case the category $\mathbf{Lens}(\mathscr{C})$.

A 2-cell in this bicategory relates two learners with different parameter sets according to the reparameterization rule in Def. 2.6, which here unpack to lens composition. The authors in (Cruttwell et al.,

[3] Modulo a difference in 2-cells: see (Cruttwell et al., 2021, Sec. 6)

2021) show that a variety of **optimizers** in machine learning (including gradient descent, momentum, and more) can be seen as 2-cells in this category.

Using the fact that **Para** is natural with respect to base change (Prop 2), Cruttwell et al. (Cruttwell et al., 2021, Sec. 3.1.) show that augmenting a reverse derivative map (Prop. 1) lifts coherently to the parameterized case via the functor

$$\mathbf{Para}(R) : \mathbf{Para}(\mathscr{C}) \to \mathbf{Para}(\mathbf{Lens}(\mathscr{C}))$$

whose unpacking we leave to the original paper.

We proceed to describe $\mathbf{Para}(\mathbf{Lens}(\mathscr{C}))$ in more detail (namely, we unpack the composition of 1-cells) by showing how it generalizes the category of learners of Fong et al (Fong et al., 2019).

Definition 2.8 (Def II.1 in (Fong et al., 2019))**.** *A **Learner** is a tuple $A \xrightarrow{(P,I,U,r)} B$ where P is a set (the parameter space) and I, U, and r are functions with types:*

$$I : P \times A \to B$$
$$U : P \times A \times B \to P$$
$$r : P \times A \times B \to A$$

The maps I, U, and r are called the implementation, update, and request maps, respectively.

It is easy to see that in this case the implementation map I corresponds to map f in $\mathbf{Para}(\mathbf{Lens}(\mathscr{C}))$, and that the update and request map can be joined into the map $Ur : P \times A \times B \to P \times A$, yielding the same backwards f^* map from Cruttwell et al's $\mathbf{Para}(\mathbf{Lens}(\mathscr{C}))$ construction.

The learners of Fong et al. (2019) form a category **Learn** where objects are sets, morphisms are learners, and the composition of the learners (P,I,U,r) and (Q,J,V,s) is $(P \otimes Q, I \fatsemi J, U \fatsemi V, s \fatsemi r)$ where:

$$(I \fatsemi J)(p,q,a) = J(q,I(p,a))$$
$$(U \fatsemi V)(p,q,a,c) = U(p,a,s(q,I(p,a),c)),V(q,I(p,a),c)$$
$$(s \fatsemi r)(p,q,a,c) = r(p,a,s(q,I(p,a),c))$$

This concrete definition of morphism composition in **Learn** is in fact the first categorical description of backpropagation, and one of the main ideas of Fong et al. (2019).

2.5.1 Learners and (Symmetric) Lenses

The connection of learners to **symmetric lenses** is examined in a follow-up paper (Fong and Johnson, 2019). A representative of a symmetric lens from A to B is a span $A \xleftarrow{(p_1,g_1)} S_1 \xrightarrow{(p_2,g_2)} B$ where (p_1,g_1) and (p_2,g_2) are asymmetric lenses. Symmetric lenses compose via pullback: the composition of $A \xleftarrow{(p_1,g_1)} S_1 \xrightarrow{(p_2,g_2)} B$ and $B \xleftarrow{(p'_1,g'_1)} S_2 \xrightarrow{(p'_2,g'_2)} C$ is the symmetric lens

$$A \xleftarrow{(\bar{p'_1},\bar{g'_1})\fatsemi(p_1,g_1)} T \xrightarrow{(\bar{p_2},\bar{g_2})\fatsemi(p'_2,g'_2)} C$$

where $S_1 \xleftarrow{(\bar{p'_1},\bar{g'_1})} T \xrightarrow{(\bar{p_2},\bar{g_2})} S_2$ is the pullback in **Set** of the cospan

$$S_1 \xrightarrow{(p_2,g_2)} B \xleftarrow{(p'_1,g'_1)} S_2$$

The authors demonstrate that we can define a faithful identity-on-objects symmetric monoidal functor from **Learn** to the category of symmetric lenses, **SLens**. This functor maps the learner $A \xrightarrow{(P,I,U,r)} B$ to the symmetric lens

$$A \xleftarrow{(k,\pi)} P \times A \xrightarrow{(\langle U,r\rangle,I)} B$$

where (k,π) is the constant complement lens (Bancilhon and Spyratos, 1981).

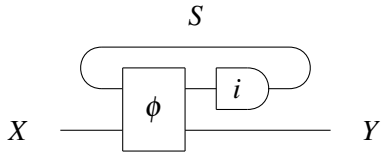
2.5.2 Learners' Languages

Spivak (Spivak, 2020b) takes an interesting approach by making a connection between **Learn** and the world of **Poly**, a category of polynomial functors in one variable. Using the fact that **Poly** is monoidal closed, this allows Spivak to interpret learners as dynamical systems. Through the language of coalgeras of polynomials, they show that the space of learners between objects A and B forms a **topos**, and consider logical propositions that can be stated in its internal language, opening up avenues to specification of *what* a learner is doing internal to the language of a learner, in the pure categorical sense.

2.6 Parameter Updates and Learning

Machine learning in the wild consists of a diverse set of techniques for training models. In this section, we examine how category theory has been applied to give a theoretical foundation for some of these techniques.

Delayed Trace and Backpropagation-Through-Time We begin with the work of Sprunger and Katsumata (2019), which answers the question of how to train models with a notion of state. More concretely, suppose instead of training a model $f : P \times X \to Y$ from examples $X \times Y$, we instead wish to learn from time-series data, where we have sequences of training points $\mathsf{List}(X \times Y)$ The idea of recurrent neural networks is essentially to model such sequences by using a stateful model:



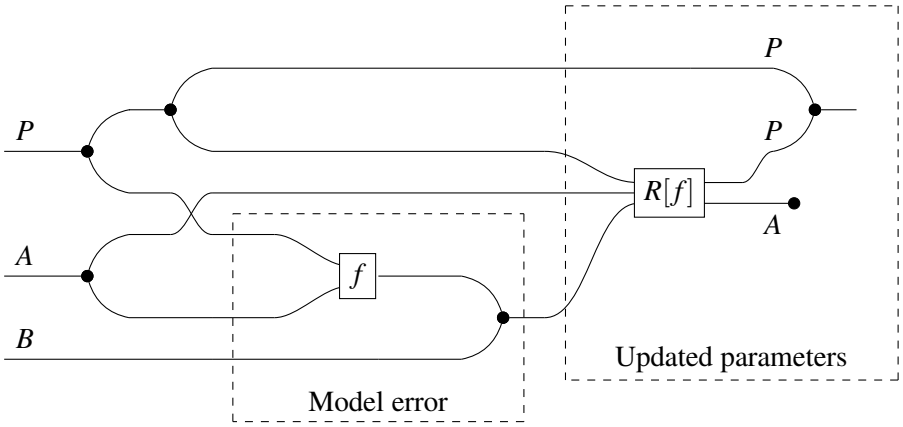
with X the model inputs, Y the predictions, and S the state. The i morphism in the diagram above is called a **delay gate**: it can be thought of as delaying its input until the next iteration of the model.

The authors construct a category of these stateful computations in which the Cartesian differential operator shares the same structure as the backpropagation through time algorithm that is used to train recurrent neural networks. Given a strict Cartesian category $\mathscr{C}$ the authors construct a double category $Dbl(\mathscr{C})$ in which 1-cells are the objects in $\mathscr{C}$ and 2-cells operate as stateful morphisms. By vertically composing these 2-cells the authors construct stateful morphism sequences. They use these sequences to define a category $\mathbf{St}(\mathbf{C})$ in which objects are $\mathbb{N}$-indexed families of $\mathscr{C}$-objects and morphisms are equivalence classes of stateful morphism sequences.

2.6.1 Generalized Optimization Algorithms

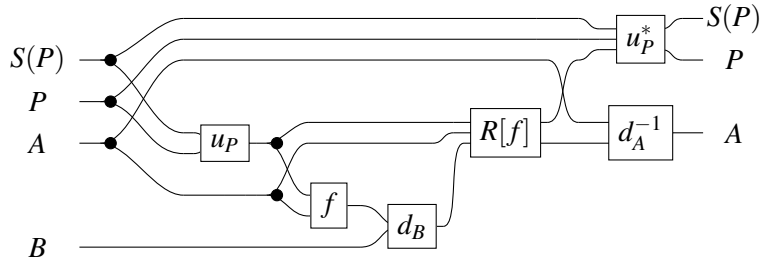
Some authors have begun to extend this generalized perspective on differentiation and derive new optimization algorithms built on top of the reverse derivative.

Reverse Derivative Ascent Wilson and Zanasi (2021) exploit the generality of the categorical reverse derivative (Section 2.2.2) to define an analogue of gradient-based methods. Although their procedure has general applications, they focus on the special case of learning Boolean circuits. For a model $f : P \times A \to B$, they give the following map to update the model's parameters:



Although not explicitly mentioned, the authors make use of lenses in their accompanying implementation, where a model is in fact a simple lens as in Definition 2.5.

Update, Displacement, and Functoriality Cruttwell et al. (2021) define a framework for categorical gradient-based learning which subsumes Reverse Derivative Ascent, as well as a number of variants of gradient descent algorithms (Ruder, 2017) including 'stateful' variants like momentum gradient descent. By way of comparison to RDA, the authors define a more general update step as follows:



While the model (f) and reverse derivative ($R[f]$) components remain in common, their approach generalises:

- **update maps** (u_P, u_P^*), defining how to update parameters given the gradient
- **displacement maps** (d_B), defining the error or distance between a model prediction and the true label

Their approach also gives conditions under which the mapping of morphisms into this category of lenses is functorial: namely, that there must exist an 'inverse displacement' map d_A^{-1}.

3 Probability and Statistics

3.1 Overview

The research in this section focuses on understanding how randomness and probability can be characterized and implemented in machine learning. This area, called **probabilistic machine learning**, includes

studying the random nature of the relationship of data we are trying to model, but also the random nature of more concrete processes, such as data sampling in our iterative learning algorithms.

Unlike the setting of neural networks, where the learning is being done on categories with appropriate differential structure, here the learning is being done on categories with appropriate probabilistic structure.

3.1.1 Applications, Successes, and Motivation

The use of category theory in probabilistic machine learning can be divided into two areas. The first one attempts to formalize and treat the notion of a **random variable** as a fundamental one, equivalent to notions such as space, group, or function. The second one attempts to use this formalization to describe how learning works in a categorical setting. The former has been extensively studied, going back to (Lawvere, 1962) and spanning dozens of papers (Corfield, 2021). The latter is less popular, but has seen a number of papers in recent years studying causality (Fong, 2013), conjugate priors (Jacobs, 2017) and general aspects of Bayesian learning (Culbertson and Sturtz, 2014, 2013).

In this paper we do not aim to examine all papers related to probability theory and category theory, but are instead focusing on the ones providing general principles most relevant to learning.

- **Synthetic probability theory.** The systemization of basic concepts from probability theory into an axiomatic framework. This enables understanding of joint distributions, marginalization, conditioning, Bayesian inverses, and more (Fritz, 2020; Fritz et al., 2020; Cho and Jacobs, 2019) purely in terms of interaction of morphisms.
- **Probabilistic programming.** The manipulation and study of programs which involve randomness (Heunen et al., 2017, 2018).
- **Probabilistic Machine Learning.** The study of updating a distribution with samples from a dataset and reasoning about uncertainty arising from noisy measurements (Culbertson and Sturtz, 2014, 2013; Jacobs, 2017).

3.1.2 Background

Given a fixed dataset, most machine learning problems reduce to optimization problems. However, solving a machine learning problem effectively requires reasoning about the source and limitations of the dataset. Essentially, there are two sources of uncertainty that separate machine learning problems from optimization problems more generally.

The first is **epistemic** uncertainty, or uncertainty that is due to hidden information. That is, information that is not available to us in the process of scientific modeling, but is available *in principle*. We can get more precise data (thus reducing epistemic uncertainty), but often we also experience experiment-dependent uncertainty: unknowns that differ each time we run the same experiment. This is called **aleatoric** uncertainty and it represents inherent uncertainty and variability in what we are trying to model.

Consider the case when we have zero aleatoric uncertainty in the problem we are trying to model: then by reducing epistemic uncertainty (by collecting more data, for instance) we can essentially reduce our machine learning problem to an optimization one. But if our problem contains aleatoric uncertainty, then collecting more data has diminishing returns: we can never perfectly deterministically model relationships that possesses inherent variability.

In most practical applications we have some form of aleatoric uncertainty and this is what probability and Bayesian inference are modelling - they shift the supervised learning from a function approximation

problem to a distribution approximation problem. In the learning theory literature this is known as **agnostic learning** (Kearns et al., 1994).

3.1.3 Big Ideas and Challenges

Many machine learning algorithms contain an irreducible aspect of randomness. The main idea of this section is that we can use category theory to reason about this randomness *internal to some category*. These advances can help us understand the high-level picture of what probabilistic learning *is* and elucidate its connections to other fields.

Nonetheless, there are many things still to do. This includes a more complete synthetic framework for reasoning about probability, making sense, for instance, in the enriched setting. Likewise, there seem to be two disjoint trends of research: Markov categories and quasi-Borel spaces. Understanding how these interact is something that has not been explored in the literature.

Going forward, we hope to see more synthetic perspectives on the primitives of Bayesian machine learning: setting priors and updating them with new data. We are also excited for more connections to be made between probabilistic perspectives and the other streams of research in this survey (Sections 2 and 4).

3.2 Categorical Probability

The field of categorical probability aims to reformulate core components of probability theory on top of category theoretic constructions. This includes things such as composition of probabilistic maps, formation of joint probability distributions, marginalization, disintegration, independence and conditionals.

Several authors, including Cho and Jacobs (2019) and Fritz (2020) claim that categories with similar monoidal and comonoidal structures serve as the right abstraction for reasoning about probability. Each of these authors introduce frameworks within which we can describe common manipulations of joint probability distributions in terms of category theoretic operations. On the other hand, Heunen et al. (2017, 2018) define the category of quasi-Borel spaces in the concrete setting of probabilistic programming. We will see that quasi-Borel spaces are an instance of Markov categories.

These constructions allow us to reason about complex probabilistic relationships in terms of algebraic axioms rather than low level analytical machinery. In this paper we will mostly be taking the vantage point of Fritz' **Markov categories** (Fritz, 2020) which generalize a number of existing constructions.

Definition 3.1 (Definition 2.1 in (Fritz, 2020))**.** *A Markov category is a symmetric monoidal category* $(\mathscr{C}, \otimes, 1)$ *in which every object* X *is equipped with a commutative comonoid structure (a comultiplication map* $\mathsf{cp} : X \to X \otimes X$ *and a counit map* $\mathsf{del} : X \to 1$*) satisfying coherence laws with the monoidal structure and where additionally the del map is natural with respect to every morphism* f*.* [4]

This seemingly opaque definition will be dissected in the following subsections. We will see how each role plays an important part which is directly related to some aspect of the concept of randomness.

The naturality of del makes the monoidal unit terminal and Markov categories semiCartesian categories. This is in contrast with CD-categories (Cho and Jacobs, 2019) which do not have this requirement. In other words, CD-categories are slightly more general and include Markov categories as *affine CD-categories*. Cho and Jacobs (2019) demonstrate that dropping the naturality condition permits a meaningful investigation of **scalars** (endomorphisms on the terminal object) and **effects** (morphisms into the terminal object).

[4] Equivalently, a Markov category is a semiCartesian category where every object is coherently equipped with the copy map.

In contrast, the naturality of del tells us that there is only one way to delete information in Markov categories. However, regardless of this condition the cp map is not necessarily natural with respect to every morphism in a Markov category. We remark on this in Section 3.2.2, where the naturality of a morphism with respect to cp implies the morphism is deterministic.

3.2.1 Distributions, Channels, Joint Distributions, and Marginalization

Markov categories are a synthetic framework for reasoning about random mappings. This can most readily be seen in the idea that a morphism $p : I \to X$ from the monoidal unit can be thought of as distribution $p(x)$ over some object X, or simply as a random element of X. To understand how this map p truly acts as a distribution, or a random element, we need to understand how it interacts with the rest of the Markov category structure.

A general morphism $f : X \to Y$ in a Markov category is often called a **channel**. The suggestive notation $f(y|x)$ is often used, denoting that the probability distribution on Y is dependent on X. The composition of p and f yields a morphism $p \fatsemi f : I \to Y$ which can be interpreted causally: first, a random element of X is generated, which is subsequently fed into f, outputting an element of Y. This allows us to interpret $p \fatsemi f$ as a random element of Y.

A morphism from the monoidal unit into the tensor of two objects $h : I \to X \otimes Y$ canonically gives us the notion of a **joint distribution** over X and Y, classically denoted as $h(x,y)$. This joint distribution can then be **marginalized** over Y to produce a distribution over X. This is done with the help of the delete map, i.e. by composition of h with $id_X \otimes \mathsf{del}_Y : X \otimes Y \to X$.

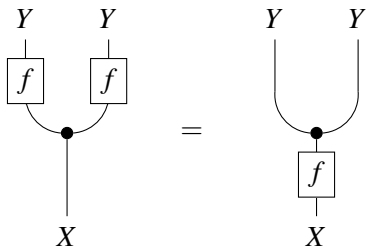
While composing the joint distribution h individually with the two projections $\mathsf{del}_X : X \otimes Y \to Y$ and $\mathsf{del}_Y : X \otimes Y \to X$ does give us two marginal distributions $I \to X$ and $I \to Y$, this is a process that cannot generally be inverted. This makes intuitive sense: the space of joint distributions contains more information than the product of their marginals and there is no bijective correspondence between $\mathscr{C}(I, X \otimes Y)$ and $\mathscr{C}(I,X) \times \mathscr{C}(I,Y)$.

This is closely related to the notion of a **deterministic morphism**, and also central to the work of (Cho and Jacobs, 2019) and (Coecke and Spekkens, 2011).

3.2.2 Deterministic Morphisms

The idea that a morphism in a Markov category is like a generalized stochastic map can be seen in the idea that we can specify what it means for a morphism to be deterministic simply by saying how it interacts with other morphisms in this Markov category.

Definition 3.2 (Def. 10.1 in (Fritz, 2020))**.** *A morphism $f : X \to Y$ in a Markov category is deterministic if it is a* cp *homomorphism:*



Recall that diagrams in this section flow from bottom to top. An intuitive way to see how f not respecting the cp structure gives rise to randomness is to think of f like rolling the dice. The process of

rolling the dice, and then copying the number we see on the top is not the same process as rolling two dice.

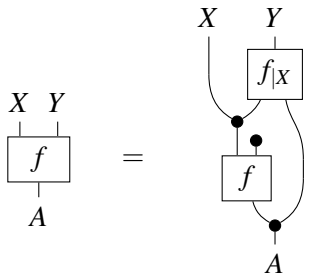
The satisfaction of this constraint would make f into a comonoid homomorphism, since del is already natural in a Markov category. A Markov category in which every morphism is deterministic is simply a Cartesian category. However, most Markov categories are not Cartesian, and this lack of determinism gives Markov categories their rich structure.

3.2.3 Conditional Probabilities

Given a joint distribution $p(x,y)$, we are often interested in computing the probability of y, *given that* x *occurred*. In other words, when conditioning y on x we are are computing the channel $p(y|x)$ such that it agrees with the information in $p(x,y)$. This is usually written as $p(x,y) = p(y|x)p(x)$, although special care has to be taken when interpreting that notation on the nose (Fritz, 2020, 2.8 Notation).

Conditionals (which Cho and Jacobs (2019) refer to as *admitting disintegration*) can be formulated in a Markov category, although not every Markov category has conditionals (Table 1).

Definition 3.3 (Def. 2.3 in (Fritz et al., 2020))**.** *Given $f : A \to X \otimes Y$ in $\mathscr{C}$, a morphism $f_{|X} : X \otimes A \to Y$ in $\mathscr{C}$ is called a* ***conditional*** *of f with respect to X if the equation*



(2)

holds. We say that $\mathscr{C}$ has conditionals provided that such a conditional exists for all $A,X,Y : \mathscr{C}$ and for all $f : A \to X \otimes Y$ in $\mathscr{C}$.

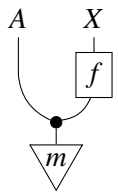
The equation 2 can be written with classical notation as $f(x,y|a) = f_{|X}(y|x,a)f(x|a)$. In a Markov category with all conditionals we can form an analogous conditional $f_{|Y}(x,y|a)$ and write Bayes' theorem in the general form:

$$f_{|X}(y|x,a)f(x|a) = f_{|Y}(x|y,a)f(y|a),$$

where the special case $A = 1$ has been treated by Cho and Jacobs (2019). Even though conditionals appear to be valuable in practice (Section 3.3.1), Fritz notes that conditioning did not end up being a relevant notion for the rest of the paper on Markov categories (Fritz, 2020, Remark 11.4).

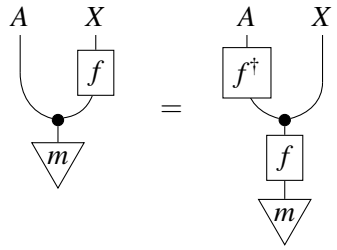
Nonetheless, conditionals enable Fritz to define a general, synthetic definition of what it means to be a **Bayesian inverse**.

Definition 3.4 (Def. 2.5 in (Fritz et al., 2020))**.** *Given two morphisms $m : I \to A$ and $f : A \to X$, a Bayesian inverse of f with respect to the prior m is a conditional of*



(3)

When the choice of the prior $m : I \to A$ is clear from the context, (Fritz et al., 2020) denotes a Bayesian inverse of f simply by $f^\dagger : X \to A$. That is, a Bayesian inverse $f^\dagger$ is defined to be the morphism satisfying the equation



(4)

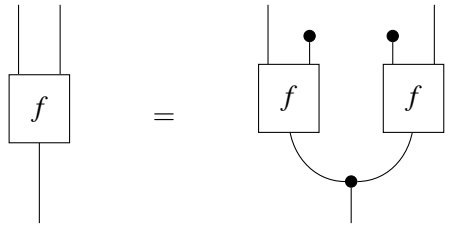
Jacobs (2017) introduces a categorical formulation of **conjugate priors**. This allows them to study the closure properties of Bayesian inversions and answer the question: "when is the posterior distribution in the same class of distributions as the prior one."

3.2.4 Independence

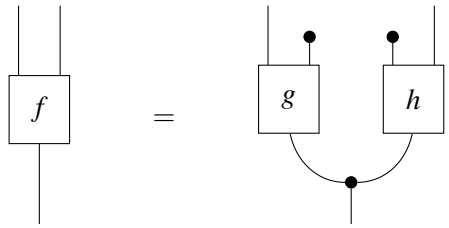
One of the most important concepts in probability theory is independence: a concept describing situations where an observation is irrelevant or redundant when evaluating a hypothesis. Typically, independence is defined in terms of random variables, or measurable functions out of a probability space, but independence can be defined abstractly in any Markov category. Fritz (2020) starts to define it in a Markov category by showing there are two equivalent characterizations.

Lemma 1 (Lemma 12.11 in (Fritz, 2020))**.** *Given a morphism $f : A \to X \otimes Y$, the following are equivalent:*

- $f(x,y|a) = f(x|a)f(y|a)$*, meaning that*



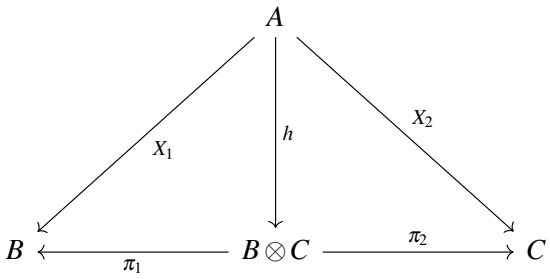
- *There are $g : A \to X$ and $h : A \to Y$ such that*



Definition 3.5 (Def. 12.12 in (Fritz, 2020))**.** *Let $\mathscr{C}$ be a Markov category. If a morphism $f : A \to X \otimes Y$ satisfies the equivalent conditions in Lemma 1, then we say that f displays conditional independence $X \perp Y \,||\, A$.*

This tells us that we get the same result if we i) generate $X \otimes Y$ from A and ii) independently generate both X and Y from A and then tensor them together.

Franz (Franz, 2002, Definition 3.4) proposes a similar definition of independence for any semiCartesian monoidal category $\mathscr{C}$. Consider the objects A, B, C and morphisms $X_1 : A \to B, X_2 : A \to C$ in $\mathscr{C}$. The morphisms X_1, X_2 are independent if there exists some morphism $h : A \to B \otimes C$ such that the following diagram commutes. Note that π_1, π_2 are the projections of the tensor product. A careful comparison of how these two definitions are related is given in (Fritz, 2020, p. 72)



While conditionals (Section 3.2.3) are often used to define independence, it can still be meaningful to introduce and work with independence in the absence of conditionals (Fritz, 2020, Ch. 12), as we've done here.

Simpson (2018) explore independence from a slightly different perspective. They define an **independence structure** on a category C to be a collection of multispans, or tuples $(X, \{f_i : X \to Y_i\})$, that form a multicategory (contain identities and closed under composition), satisfy a number of coherence properties, and contain every singleton family $(X, \{f : X \to Y\})$.

3.2.5 Examples of Markov Categories

There are a large number of examples of Markov categories, and it is possible to organize them in many families. Firstly, every Cartesian category is a Markov category in a trivial way, in essence containing only the deterministic Markov morphisms. But many interesting Markov categories are not Cartesian and contain many non-deterministic morphisms. Two examples are **Gauss**, the category of Gaussian conditionals, and **FinStoch**, the category of Markov kernels between finite sets. These are some of the Markov categories that do *not* arise as Kleisli categories. The most common way to construct Markov categories is as Kleisli categories of various affine monoidal monads on various base categories. The converse of this result is only partially understood and is described in (Fritz et al., 2020, Sec. 3.2).

Proposition 3 (Prop. 3.1 in (Fritz et al., 2020))**.** *Let $\mathscr{C}$ be a category with finite products. Let D be a commutative monad on $\mathscr{C}$ with $D(1) \cong 1$. Then the Kleisli category $Kl(D)$ is a Markov category.*

The simplest example is that of **Set** with the distribution monad [5], thoroughly explored in (Fritz, 2009). See Table 1 for more examples:

[5] Also called the **finitary Giry monad** or the **convex combination monad**.

Base category	Monad D	$Kl(D)$	Has all conditionals?	Reference
Set	Distribution	- [6]	?	(Fritz, 2009, Definition 3.3)
Meas	Giry	**Stoch**	? (Fritz, 2020, Ex. 11.7)	(Fritz, 2020)
FinMeas	Giry	**FinStoch**	✓(Fritz, 2020, Ex. 11.6)	(Fong, 2013, p. 31)
CGMeas	Giry	**CGStoch**	✓(Fong, 2013, Def. 3.4)	(Fong, 2013, p. 31)
Pol	Giry	**BorelStoch**	✓(Fritz et al., 2020, Ex. 2.4)	(Fritz et al., 2020, Example 3.2)
QBS	-[6]	-[6]	?	(Heunen et al., 2017)
CHaus	Radon	- [6]	X (Fritz, 2020, Ex. 11.4)	(Fritz, 2020, Section 5)

Table 1: A bird's-eye view of some Markov categories that arise as Kleisli categories of various monads.

One of the most common and widely used examples is **Stoch** and many of its subcategories. The category **Stoch** naturally arises as the Kleisli category of the **Giry Monad**, which is a monad on the category **Meas** of measurable spaces.

Definition 3.6. *Measurable spaces and measurable functions form a category* **Meas**. *The objects in* **Meas** *are pairs (A,Σ_A), where Σ_A is a σ-algebra over A. A morphism from (A,Σ_A) to (B,Σ_B) in* **Meas** *is a measurable function f such that for any $\sigma_B \in \Sigma_B$, $f^{-1}(\sigma_B) \in \Sigma_A$.*

The category **Meas** is the base of the **Giry** monad (Giry, 1982): an affine symmetric monoidal monad that sends a measurable space X to the measurable space of probability measures over X. We skip the complete definition and instead simply define **Stoch** as the category with measurable spaces as objects and **Markov kernels** as morphisms.

Definition 3.7. *A **Markov kernel** between the measurable space (A,Σ_A) and the measurable space (B,Σ_B) is a function $\mu : A \times \Sigma_B \to [0,1]$ such that:*

- *For all $\sigma_b \in \Sigma_B$, the function $\mu(\_,\sigma_b) : A \to [0,1]$ is measurable.*
- *For all $x_a \in A$, $\mu(x_a,\_) : \Sigma_B \to [0,1]$ is a probability measure on (B,Σ_B). In particular:*

$$\mu(x_a,B) = 1 \qquad \mu(x_a,\varnothing) = 0$$

For example, a Markov Kernel between the one-point set and the measurable space (A,Σ_A) is just a probability measure over (A,Σ_A). We define the composition of the Markov kernels $\mu : A \times \Sigma_B \to [0,1]$ and $\mu' : B \times \Sigma_C \to [0,1]$ to be the following, where $x_a \in A$ and $\sigma_c \in \Sigma_C$:

$$(\mu \fatsemi \mu')(x_a,\sigma_c) = \int_{x_b \in B} \mu'(x_b,\sigma_c) d\mu(x_a,\_)$$

The identity morphism at (A,Σ_A) is δ where:

$$\delta(x_a,\sigma_a) = \begin{cases} 1 & x_a \in \sigma_a \\ 0 & x_a \notin \sigma_a \end{cases}$$

The tensor product of the Markov Kernels $\mu : A \times \Sigma_B \to [0,1]$ and $\mu' : C \times \Sigma_D \to [0,1]$ in **Stoch** is the following Markov Kernel, where $\Sigma_B \otimes \Sigma_D$ is the product σ-algebra:

$$(\mu' \otimes \mu) : (A \times C) \times (\Sigma_B \otimes \Sigma_D) \to [0,1]$$
$$(\mu' \otimes \mu)((x_a,x_c),\sigma_b \times \sigma_d) = \mu(x_a,\sigma_b) * \mu(x_c,\sigma_d)$$

[6] This construction does not have a particular name.

Originally introduced by Lawvere (1962), the category **Stoch** is the paradigmatic example of a Markov category (Fritz, 2020, Ch. 4). One of the most interesting aspects of **Stoch** is that many of the most important operations in probability, including marginalization, disintegration, and independence, can all be characterized internally to **Stoch**.

While one of the most used constructions, **Stoch** admits a few pathological examples (Fong, 2013, p. 26), and it is not known whether it has conditional probabilities in general (Fritz, 2020, Ex. 11.7). Furthermore, the induced functor **Meas** $\to$ **Stoch** [7] is not faithful (Fritz, 2020, Ex. 10.4.). Fritz notes that **Stoch** might not be actually the "best" Markov category for measure-theoretic probability (Fritz, 2020, p. 31). These are the reasons that many authors often work with its subcategories. Fong (2013) and Culbertson and Sturtz (2014) work with **CGStoch**, while Fritz et al. (2020) works with, among others, **BorelStoch** and **FinStoch**.

The subcategory **CGStoch** of **Stoch** has countably generated measurable spaces as objects and Markov kernels over perfect probability measures as morphisms. The subcategory **BorelStoch** of **CGStoch** adds the additional condition of separability and is defined as the Kleisli category of the Giry monad on **Pol**, the category of Polish spaces and measurable maps (Doberkat, 2004). Both **CGStoch** and **BorelStoch** have all conditionals (Fong, 2013; Faden et al., 1985). The restriction of **Stoch** to the finite case as a subcategory **FinStoch** also yields a category with conditionals (Fritz, 2020, Ex. 11.6)

3.2.6 Cartesian Closedness and Quasi-Borel Spaces

When considering probabilistic programs, the distributions we manipulate are not arbitrary, but come from a particular random source. Similarly, in statistics and probability theory, we focus primarily on random variables over some fixed global sample space rather than arbitrary probability measures (Heunen et al., 2018, Sec. 2). Furthermore, in the context of probabilistic programming it is often desirable to reason about the space of probabilistic mappings internal to the probabilistic category at hand. However, while **Meas** is symmetric monoidal, it is not Cartesian closed, since the space of measurable maps into spaces of measurable maps is not always measurable. Culbertson and Sturtz (2013) attempts to address this problem by simply equipping **Meas** with a different monoidal product.

Heunen et al. (2017) use a different strategy, and instead generalize measurable spaces to **quasi-Borel spaces QBS**, which are Cartesian closed (Heunen et al., 2017, Prop. 18).

Definition 3.8 (Def. 7 in (Heunen et al., 2017))**.** *A **quasi-Borel** space is a set X, its carrier, together with a subset M_X of the function space $\mathbb{R} \to X$ satisfying the following conditions:*

- ***Closure with respect to precomposition with measurable morphisms.*** *For every $\alpha \in M_X$ and every measurable $f : \mathbb{R} \to \mathbb{R}$ the following holds: $f \fatsemi \alpha \in M_X$.*
- ***Inclusion of constants.*** *For every constant $\alpha : \mathbb{R} \to X$ we have that $\alpha \in M_X$.*
- ***Closure under gluing with disjoint Borel domains.*** *For every countable, measurable partition $\mathbb{R} = \uplus_{i:\mathbb{N}} S_i$ and every sequence $(\alpha_i)_{i:\mathbb{N}}$ in M_X, we have that $\beta \in M_X$ and is defined as $\beta(r) := \alpha_i(r)$ for all $r : S_i$.*

There are a number of ways to characterize quasi-Borel spaces, including as structured measurable spaces (Heunen et al., 2017, III.B.1), or as a conservative extension of standard Borel spaces that supports simple type theory (products, coproducts and function spaces) (Heunen et al., 2017, IV.). Heunen et al. (2017) demonstrate the use of quasi-Borel spaces by showing that a well-known construction of probability theory involving random functions gains a clear expression. They also generalize de Finetti's theorem to quasi-Borel spaces.

[7] This is the example of a canonical functor $\mathscr{C} \to Kl(D)$ for some monad D defined on $\mathscr{C}$.

3.3 Causality and Bayesian Updates

There are a wide variety of category theoretic approaches to simple Bayesian learning. For example, Fong (2013) uses a graphical framework to generalize Bayesian networks. He first defines a **causal theory** to be a strict symmetric monoidal category generated by a directed acyclic graph and equipped with a comonoidal structure on each object. Next, he shows how a functor F from a causal theory with vertex set V into **Stoch** is equivalent to a Bayesian network with variables in V: given some edge $A \to B$ in the causal theory, the Markov Kernel that F maps this edge to encodes the probabilistic relationship between A and B. Forward (aka predictive) inference is then mediated by the compositions of morphisms in the DAG and their images under F. This construction is very general, and **Stoch** can be replaced with any other Markov category. Howver, Fong does not describe how causal theories could be used for backwards inference or how they can be learned from data.

Jacobs (2018) takes a slightly different perspective and generalizes Bayesian networks with a construction similar to Gavranovic (2019)'s formulation of a neural network (see Section 4.3):

Proposition 4. *Let G be a directed acyclic multi-graph and D some probability monad. Then the conditional tables of a Bayesian Network are given by a strong monoidal functor from the free category of G to the Kleisli category of D.*

Jacobs (Jacobs, 2018) describes a similar framework for Bayesian updates to a prior over a joint distribution. He focuses on multinomial distributions and treats multisets with n unique elements as the parameter vectors for n-category multinomial distributions. He describes two natural transformations out of the multiset monad $\mathscr{M}$ which commute with an expected value operation:

- The normalization function $l_n : \mathscr{M}(n) \to \mathscr{D}(n)$, which maps a multiset in $\mathscr{M}(n)$ to its corresponding maximal likelihood estimate. This estimate is a multinomial distribution in $\mathscr{D}(n)$, where $\mathscr{D}$ is the distribution monad.
- The Dirichlet distribution function $Dir_n : \mathscr{M}(n) \to \mathscr{G}(\mathscr{D}(n))$, which maps a multiset in $\mathscr{M}(n)$ to the Dirichlet distribution parameterized by that multiset. This distribution is a probability measure over multinomial distributions and is therefore an element of $\mathscr{G}(\mathscr{D}(n))$, where $\mathscr{G}$ is the Giry monad.

Jacobs characterizes the normalization and Dirichlet distribution natural transformations as frequentist and Bayesian learning procedures respectively. While an update to the parameters of the Dirichlet distribution smoothly adjusts the distribution over multinomials, applying the same procedure to the output of the normalization function will cause the distribution to collapse as it overcorrects to the new observation.

These works also hint at how categorical probability can serve as a basis for a structural perspective on machine learning. Cho and Jacobs (2019) use their generalized string diagram formulation of Bayesian inversion to step through a small example of naive Bayes classification: they first disintegrate channels from a table of data, and then invert the combined channels to yield a classification. It's worth noting how different this formulation is from the optics-based perspective on learning described in Section 2. Whereas the optics papers abstract away the objective function and focus on the mechanics of model updates, Cho and Jacobs' objective is inextricably tied to the marginalization, inversions, and disintegrations in the learning procedure.

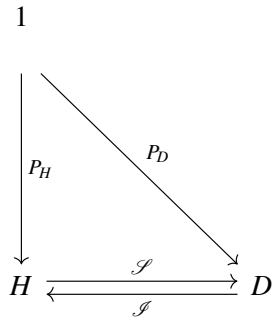
3.3.1 Bayes Law; Concretely

While Bayes law was described abstractly in Section 3.2.3 and grounded in a Markov category with conditionals, in this subsection we describe the more practical aspect of Bayes law in the work of Culbertson and Sturtz (2013, 2014). Appearing six years before Markov categories, the authors work

within the concrete setting of **CGStoch**, permitting them to talk about conditionals, unlike the more general setting of **Stoch**.

The authors focus entirely on supervised learning but note that the general framework applies to any Bayesian machine learning problem. They also develop a categorical framework for updating a prior with data and study stochastic processes using functor categories, demonstrating the Kalman filter as an archetype for the hidden Markov model.

This enables Culbertson and Sturtz (2013) to define a Bayesian model to be a diagram:



- **Hypothesis space**: A measurable space H that serves as the space of potential models.
- **Data space**: A measurable space D that serves as the space of possible experiment outcomes.
- **Prior probability measure**: An arrow $P_H : 1 \to H$.
- **Sampling distribution**: An arrow $\mathscr{S} : H \to D$ that relates the models in H to distributions over the data space D.
- **Inference Map**: An arrow $\mathscr{I} : D \to H$ that relates new data observations to posterior probabilities over H.

The authors construct $\mathscr{I}$ such that for $\sigma_D \in D, \sigma_H \in H$:

$$\int_{x_D \in \sigma_D} \mathscr{I}(x_D, \sigma_H) dP_D = \int_{x_H \in \sigma_H} \mathscr{S}(x_H, \sigma_D) dP_H$$

That is, $\mathscr{I}$ is the regular conditional probability constructed from the joint distribution over $H \times D$ that P_H and $\mathscr{S}$ define. Given a new "data observation" (which the authors represent with a probability measure μ over D, or an arrow $1 \to D$), the inference map $\mathscr{I}$ defines the posterior probability over H to be:

$$\hat{P_H}(\sigma_H) = \int_{x_D \in D} \mathscr{I}(x_D, \sigma_H) d\mu$$

Culbertson et al's construction iterates the following process for each new data point $\mu : 1 \to D$:

1. Set $\hat{P_H}(\sigma_H) = \int_{x_D \in D} \mathscr{I}(x_D, \sigma_H) d\mu$.
2. Set P_H to $\hat{P_H}$.
3. Define $\mathscr{I}$ from P_H and $\mathscr{S}$.

Culbertson et al. also describe how this process can work in the Eilenberg-Moore category of the Giry nonad G, which they dub the category of **decision rules**. As the Kleisli category of the Giry Monad, **Stoch** embeds into this category. The objects in this category are G-algebras, or pairs of a measurable

space X and a measurable map $\alpha : GX \to X$. Since GX is the space of probability distributions over X, we can think of α as a decision rule that collapses a distribution over X to a single value.

In a machine learning application we would expect that the elements of the "hypothesis space" H are maps over the data space D. The cleanest way to represent such maps would be as exponential objects. However, **Meas** is not closed over the Cartesian product.

In order to solve this, Culbertson and Sturtz (2013) define a new monoidal product such that **Meas** becomes closed. This is in contrast to the solution in "A Convenient Category for Higher-Order Probability Theory" (Heunen et al., 2017), where the authors choose to instead redefine **Meas** itself. Under the new tensor product in Culbertson and Sturtz (2013), the σ-algebra on $X \otimes Y$ is the largest σ-algebra such that the **constant graph functions** $\Gamma_{\bar{y}} : X \to X \otimes Y$ and $\Gamma_{\bar{x}} : Y \to X \otimes Y$ that respectively send $x \in X$ to $(x, \bar{y})$ and $y \in Y$ to $(\bar{x}, y)$ are measurable.

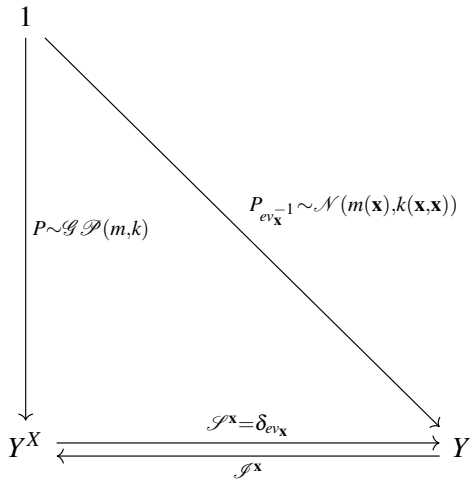
Culbertson et al. also demonstrate that when we equip **Meas** with this monoidal product, it becomes a symmetric monoidal closed category such that the constant graph functions:

$$\Gamma_{\bar{f}} : X \to X \otimes Y^X$$
$$\Gamma_{\bar{f}}(x) = (x, \bar{f})$$

and the evaluation maps

$$ev_{X,Y} : X \times Y^X \to Y$$
$$ev_{X,Y}(x, f) = f(x)$$

are both measurable. This allows the authors to define Bayesian models over function spaces. For example, suppose we have a Gaussian Process $\mathscr{GP}(m,k)$, where $m \in Y^X$ and k is a kernel function $k : X \times X \to \mathbb{R}$. Then the following diagram shows how the composite of a prior distribution over hypotheses that this Gaussian Process defines and a sampling distribution $\mathscr{S}^{\mathbf{x}} = \delta_{ev_{\mathbf{x}}}$ defined by an observation $\mathbf{x}$ defines a data distribution over Y and an inference map from Y to Y^X.



In this framework, an arrow of the form $1 \to Y^X$ is a stochastic process, and an arrow of the form $Z \to Y^X$ is a parameterized stochastic process.

Shiebler (2020a) also explores a categorical perspective on learning algorithms. Like Culbertson and Sturtz (2013), Shiebler focuses on parameterized stochastic processes, which he represents as morphisms in a

category. Shiebler explores two ways in which stochastic processes can compose: a co-Kleisli composition in which randomness is shared and a Para composition in which each stochastic process is equipped with its own probability space.

Unlike Culbertson et al.'s framework, Shiebler's framework is built on a frequentist perspective. Shiebler particularly focuses on the composition of likelihood functions and the derivation of loss functions from likelihoods. To do this Shiebler constructs a monoidal semicategory of generalized likelihood functions, and he defines a strict monoidal semifunctor from a category of stochastic processes into this category. Shiebler then uses this construction to develop backpropagation functors into **Learn** in which the error function is derived from the likelihood.

3.4 Optics for Probability

Some authors have begun to leverage similar strategies to those described in Section 2 to study lenses and optics in probability.

For example, Smithe (2020) leverages Cho and Jacobs (2019)'s synthetic approach to Bayesian inversion to demonstrate that Bayesian updates compose optically. Since his construction relies on the Cartesian-closedness of the base category, Smithe's primary example base category is the category of quasi-Borel spaces **QBS** (Heunen et al., 2017) rather than the category **Meas** of measurable spaces.

Given a probability monad $\mathscr{P}$ on **QBS**, Smithe writes $Kl(\mathscr{P})$ for its Kleisli category and constructs the $Kl(\mathscr{P})$-state-indexed category $Stat : Kl(\mathscr{P})^{op} \to \mathbf{QBS}-\mathbf{Cat}$, where $\mathbf{QBS}-\mathbf{Cat}$ is the category of **QBS**-enriched categories. $Stat$ is a functor that maps $X \in \mathbf{QBS}$ to the category $Stat(X)$ with the same objects as **QBS**. The morphisms in $Stat(X)(A,B)$ are stochastic channels $\mathscr{P}X \to \mathbf{QBS}(A, \mathscr{P}B)$. Smithe then applies Spivak's Grothendieck lens construction (Spivak, 2020a) to $Stat$ to form the category $\mathbf{GrLens}_{\mathbf{Stat}}$ in which objects are pairs of objects of $Kl(\mathscr{P})$ and morphisms are $Stat$-lenses, or elements of the set:

$$Kl(\mathscr{P})(X,Y) \times \mathbf{QBS}(Kl(\mathscr{P})(I,X), Kl(\mathscr{P})(B,A))$$

Smithe then demonstrates that $Stat$-lenses compose as optics and uses this fact to define a category of lenses, **BayesLens**, within which $\mathbf{GrLens}_{\mathbf{Stat}}$ is a full subcategory. In particular, every pair of a stochastic channel and its inverse (as in (Cho and Jacobs, 2019)) forms a Bayesian lens. Smithe calls such lenses **exact Bayesian lenses**. Smithe's main result is essentially that the composition of two exact Bayesian lenses yields another exact Bayesian lens. That is, Bayesian inversion is almost-everywhere preserved upon lens composition.

One of the benefits of this perspective is that Smithe can study the lawfulness of Bayesian lenses, similarly to Fong and Johnson's exploration of the lawfulness of Learners as Lenses (Fong and Johnson, 2019). Due to the inherent uncertainty in the processes that Bayesian lenses model, it is not surprising that none of the three laws (GetPut, PutGet, PutPut) hold in general. In particular, GetPut and PutGet only hold with respect to states (as in (Cho and Jacobs, 2019)) since a Bayesian update after observing data that exactly matches our prediction does not result in a change to our prior.

3.5 Functorial Statistics

Some authors have also used functoriality to explore how the different pieces of statistical learning systems can fit together. McCullagh (2002) attempts to unearth the invariants at the heart of statistical modeling. He does so by forming categories over which we can define statistical designs and model parameters as functors such that statistical models form natural transformations between them. The

commutativity of these natural transformations enforces that parameter maps maintain their structure independent of sample space transformations and that the "meaning" of a parameter does not change in response to modifications of the statistical design.

Furthermore, Allison (2003) defines a system for representing components and categories of statistical models as types and classes. He uses this system to identify and factor out the common components between different types of algorithms, such as mixture modeling and decision trees. He also describes how recombinations of these common components can lead to new algorithms, although he doesn't explore this idea in detail.

4 Invariant and Equivariant Learning

4.1 Overview

The research in this section focuses on the symmetry-preserving properties of machine learning algorithms. The authors whom we cite use a variety of tools for exploring the relationships between transformations of datasets and the outputs of machine learning models that run on those datasets. Many of these tools are explicitly category theoretic, such as functors and natural transformations.

4.1.1 Applications, Successes, and Motivation

The authors in this section discuss a variety of different kinds of machine learning algorithms that are used widely in practice:

- **Clustering** (Section 4.2.1): Clustering algorithms group points in the same dataset together. Social networks use clustering algorithms to identify users who share interests or social connections (Satuluri et al., 2020).
- **Manifold Learning** (Section 4.2.5): Manifold learning algorithms map the points in a dataset to low-dimensional embeddings in $\mathbb{R}^n$, which can then be used as features for machine learning models. Manifold algorithms are also commonly used along with nearest neighbor search algorithms to power recommendation engines (Shiebler et al., 2018).
- **Convolutional Neural Networks** (Section 4.4): Convolutional neural networks process image data in a way that exploits the position invariance inherent to most image processing tasks. These are used in basically every modern image processing application (Linsley et al., 2019).
- **Graph Neural Networks** (Section 4.4): Graph neural networks process graph data in a way that exploits both node features and internode connectivities. Graph neural networks are used widely for traffic forecasting (Jiang and Luo, 2021).

4.1.2 Background

Several of the authors whom we cite in this section rely heavily on the vocabulary of topological data analysis, including simplicial complexes and filtrations. We recommend Chazal (Chazal and Michel, 2017) for a good introduction to these topics.

Briefly, a **finite simplicial complex** is a family of finite sets that is closed under taking subsets. The finite sets in a simplicial complex are called **faces**, and the ***n*-simplices** are subcomplexes that contain all of the subsets of a single $(n+1)$-element face in the complex. For example, the 0-simplices (also called vertices) are points, the 1-simplices are pairs of points, the 2-simplices are triples of intersecting

1-simplices that we can visualize as triangles, etc. Note that any n-simplex in a simplicial complex is completely determined by the set of vertices that span it.

A **simplicial map** between a simplicial complex S_X with vertices X and a simplicial complex S_Y with vertices Y is a function $f : X \to Y$ such that if $x_1, x_2, \cdots, x_n$ span an n-simplex in S_X, then $f(x_1), f(x_2), \cdots, f(x_n)$ span an m-simplex in S_Y where $m \leq n$. Simplicial complexes and simplicial maps form a category **SCpx**.

Given a metric space (X, d_X) and a choice of $\delta \in \mathbb{R}_{\geq 0}$, the **$\delta$-Vietoris-Rips complex** is the simplicial complex whose n-simplices are the n-element subsets of X with all pairwise distances no greater than δ. Note that if $\delta \leq \delta'$, then the set of n-simplices of the δ-Vietoris-Rips complex is a subset of the set of n-simplices of δ'-Vietoris-Rips complex. A sequence $\delta_1 \leq \delta_2 \leq ...$ induces a sequence of Vietoris-Rips complexes, each of which contains the simplices of the previous complex. Such a sequence is called a **filtration**.

4.1.3 Big Ideas and Challenges

The main idea of this section is that many powerful machine learning algorithms are invariant or equivariant to certain kinds of dataset transformations. Understanding these properties allows us to better understand how algorithms separate signal from noise. By characterizing an algorithm as a functor or natural transformation we can encode these invariances and equivariances in the morphisms of the source and target category.

However, for more sophisticated algorithms these invariance and equivariance properties often hold only approximately. They may fall apart completely at the edges (like convolution's position invariance) or require properties that hold only in special cases (like the invertibility of the data matrix in linear regression). This makes it difficult to formalize these properties in the language of functors and natural transformations. The authors in this section use a number of strategies to get around this challenge, including approximate commutation (Cohen and Welling, 2016; Guss and Salakhutdinov, 2019) and exploiting symmetry in the loss function rather than the algorithm output (Shiebler, 2021b).

4.2 Functorial Unsupervised Learning

An unsupervised learning algorithm is any algorithm that aims to extract insights from data without explicit supervision. The class of unsupervised algorithms is much more general than that of supervised algorithms, but most unsupervised algorithms operate by doing one or both of the following:

- Determining the shape of the probability distribution that the data was drawn from.
- Estimating a low dimensional manifold that the data is assumed to lie upon.

In order for an unsupervised algorithm to be useful, the properties of the lower dimensional manifold or probability distribution that the algorithm uses to describe the observed data must be somewhat in line with the structure of that data. One way to formalize this is to cast these algorithms as various kinds of functors and characterize this property in term of functoriality.

4.2.1 Functorial Clustering

One of the most common classes of unsupervised learning algorithms is clustering algorithms. A **clustering** of a metric space (X, d_X) is essentially a partitioning of X such that the points x, x' are more likely to be in the same partition if $d(x, x')$ is small.

Algorithm	Description	Produces non-overlapping clusters?	Is functor from **Met**?	Is functor from $\mathbf{Met}_{inj}$?
Single Linkage	Connected components of Rips complex	Yes	Yes	Yes
Robust Single Linkage	Connected components of rescaled Rips complex	Yes	No	Yes
Maximal Linkage	Simplices of Rips complex	No	Yes	Yes
KMeans	Centroids that minimize within-cluster variance	Yes	No	No

Table 2: Clustering algorithms

Definition 4.1. *Given a partition $\mathscr{P}_X$ of the set X and a partition $\mathscr{P}_Y$ of the set Y, a **refinement-preserving map** from $\mathscr{P}_X$ to $\mathscr{P}_Y$ is a function $f : X \to Y$ such that for any $S \in \mathscr{P}_Y$, there exists some $S' \in \mathscr{P}_X$ with $f(S') \subseteq S$.*

Carlsson and Mémoli (2008, 2013) describe clustering algorithms as functors from one of the following categories of metric spaces into the category **Part** of partitions and refinement-preserving maps:

- **Met**: The category of metric spaces and **non-expansive map**s between them. A non-expansive map between the metric spaces (X, d_X) and (Y, d_Y) is a function $f : X \to Y$ such that for $x_1, x_2 \in X$, $d_X(x_1, x_2) \geq d_Y(f(x_1), f(x_2))$.
- $\mathbf{Met}_{inj}$: The category of metric spaces and injective non-expansive maps between them.
- $\mathbf{Met}_{isom}$: The category of metric spaces and isometries (distance-preserving isomorphisms) between them.

Note that $\mathbf{Met}_{isom}$ is a subcategory of $\mathbf{Met}_{inj}$, which is in turn a subcategory of **Met**. An example clustering functor from **Met** to the category of partitions and refinement-preserving maps is the following:

Definition 4.2. *The **single linkage at distance** δ functor maps a metric space (X, d_X) to the partition of X such that the points x_1, x_n are the same partition only if there exists some chain of points $x_1, x_2, ..., x_n$ such that $d_X(x_i, x_{i+1}) \leq \varepsilon$.*

The clusters defined by single linkage clustering are the connected components of the Vietoris-Rips complex, and can be viewed as a discrete approximation of the path components in a topological space.

A particularly important result in clustering theory is the Kleinberg **Impossibility Theorem** (Kleinberg, 2003). This Theorem states that it is impossible to define a clustering algorithm that satisfies all three of the following properties:

- **Scale Invariance**: The clustering function should not be sensitive to the units of measurement. Any algorithm that uses a distance hyperparameter (such as single linkage clustering) fails to satisfy scale invariance.
- **Richness/Surjectivity**: The clustering function should be able to produce any clustering, given an appropriate distance function. Note that a clustering algorithm that satisfies this condition is morally similar to a classification algorithm that can shatter any set of points (i.e infinite VC

dimension (Vapnik and Chervonenkis, 2015)). Any algorithm that requires a pre-set number of clusters (such as K-means) fails to satisfy this condition.

- **Consistency**: Shrinking the distances between points in the same cluster and increasing the distances between points in different clusters does not change the clustering. Any centroid-based algorithm (such as Mixture of Gaussians) fails to satisfy this condition.

One strategy for getting around this restriction is to define **hierarchical clustering algorithms** that generate a series of clusterings, each at a different scale (thereby relaxing scale invariance). There are a number of formalizations of hierarchical clustering algorithms. For example, Carlsson and Mémoli (2008) define a hierarchical clustering algorithm to be a functor from a category of metric spaces to a category of **persistent sets** and **persistence-preserving maps**

Definition 4.3. *A persistent set is a pair (X,θ) where X is a finite set and θ is a function from the non-negative real line $[0,\infty)$ to the set of partitions of X so that the following properties hold:*

1. *If $r \leq s$ then $\theta(r)$ refines $\theta(s)$.*
2. *For any r, there is a number $\varepsilon > 0$ so that $\theta(r') = \theta(r)$ for all $r' \in [r, r+\varepsilon]$.*

Definition 4.4. *A **persistence-preserving map** is a function $f : (X,\theta) \to (Y,\eta)$ such that any partitioning that f^{-1} induces on X is a refinement of the partitioning θ induces.*

We can alternatively define persistent sets as functors from the non-negative real line to the category in which objects are partitions of X and morphisms are refinement-preserving maps. In this formulation persistence-preserving maps are natural transformations between persistent sets.

While a variety of agglomerative hierarchical clustering algorithms are functorial over $\mathbf{Met}_{isom}$, a much smaller set of algorithms is functorial over $\mathbf{Met}_{inj}$, and the following is the unique well-behaved functor over **Met**.

Definition 4.5. *The **single linkage functor** $\mathscr{SL}$ maps a metric space (X,d_X) to the persistent set (X,θ) where for some $\varepsilon \in [0,\infty)$, the points x_1, x_n are in the same partition in $\theta(\varepsilon)$ only if there exists some chain of points $x_1, x_2, ..., x_n$ such that $d_X(x_i, x_{i+1}) \leq \varepsilon$.*

There are a number of major issues with the single linkage clustering algorithm that make it perform poorly in practice. For example, it is extremely sensitive to noise points, which can fall between distinct clusters and cause them to be erroneously bridged. For this reason, practical applications of single linkage clustering generally include pre/post-processing stages that rescale distances or remove points to reduce the impact of noise.

For example, Wishart (1969) proposes first defining a density estimate over the space and then removing points that appear in low-density regions. There are several ways to implement this idea. For example, Chaudhuri and Dasgupta (2010) introduce the **Robust Single Linkage** algorithm, which identifies the low-density points to be points with fewer than k neighbors within an open ball of radius ε.

In another paper Carlsson and Mémoli (2013) explore how we can define a broad family of clustering algorithms in $\mathbf{Met}_{inj}$ that factor through single linkage clustering. Since maps in $\mathbf{Met}_{inj}$ must be injective, clustering functors over $\mathbf{Met}_{inj}$ can be sensitive to the number of points in a region, such as Chaudhuri and Dasgupta (2010)'s Robust Single Linkage approach . Such a density sensitive mapping would not be functorial over **Met** because a morphism in **Met** may collapse multiple points into the same point.

The authors note that there are many uninteresting functorial mappings from $\mathbf{Met}_{inj}$ to the category of partitions and refinement-preserving maps. They define an additional property, **excisiveness**, to restrict to

a more interesting class of maps. An excisive clustering functor is one that is idempotent in that applying the functor to any of the partitions it forms will not further subpartition that partition.

Carlsson and Memoli's primary result is an explicit generative modeling framework for expressing any excisive clustering functor from $\mathbf{Met}_{inj}$ to the category of partitions and refinement-preserving maps. This framework can represent any such functor as a finite collection of finite metric spaces Ω. The represented functor then maps the points x, x' in the metric space X to the same partition if and only if there exist:

- a sequence of $k+1$ points $x, x_1, ..., x_{k-1}, x' \in X$
- a sequence of k metric spaces $\omega_1, ..., \omega_k \in \Omega$
- a sequence of k morphisms $f_1, ..., f_k$ from $f_i : \omega_i \to X$ such that there exist points $(\alpha_i, \beta_i) \in \omega_i$ where $f_i(\alpha_i) = x_{i-1}, f_i(\beta_i) = x_i$

The single linkage functor $\mathscr{SL}(\delta)$ at distance δ is then represented by the collection $\Omega = \{\Delta_2(\delta)\}$ where $\Delta_2(\delta)$ is the finite 2-point metric space where the points are separated by distance δ.

This functor is fundamental in the sense that any clustering functor F expressible by this framework factors through $\mathscr{SL}(\delta)$ such that $F = G \fatsemi \mathscr{SL}(\delta)$ where G is a functor that commutes with the forgetful functor from metric spaces to **Set** (i.e. it maps the metric space (X, d_X) to (X, d'_X)). It is worth noting that "transform the distance metric and then apply single linkage clustering" is a powerful recipe for noise-resistant clustering algorithms, such as DBSCAN (Ester et al., 1996).

4.2.2 Functorial Overlapping Clustering

Culbertson et al. (2016) use a different approach from Carlsson and Mémoli (2013, 2008) and study clustering algorithms that can produce overlapping clusters. This produces another lever to mitigate the negative impact of chaining that occurs in single linkage clustering: since the relation that groups points into clusters does not need to be transitive, the algorithm can enforce maximum distance constraints on the points in the same clusters. The authors particularly focus on **non-nested flag covers**, or non-nested coverings whose associated abstract simplicial complexes are **flag complexes**, or simplicial complexes that can be expressed as the cliques of its 1-skeleton.

Definition 4.6. *In the category* **Cov***, objects are covers $(X, \mathscr{C}_X)$ and the morphisms between $(X, \mathscr{C}_X)$ and $(Y, \mathscr{D}_Y)$ are* ***consistent maps****, or functions $f : X \to Y$ such that $\mathscr{C}_X$ is a refinement of $f^{-1}(\mathscr{D}_Y)$.*

Note that refinement-preserving maps are just consistent maps between partitionings, and that **Part** is a subcategory of **Cov**. One particularly important kind of flag cover is the following:

Definition 4.7. *Given a symmetric, reflexive relation R on the set X, the set of* ***maximally linked*** *subsets of X consists of all $S \subseteq X$ where (1) xRy for all $x, y \in S$ and (2) S is not properly contained in any subset of X satisfying (1).*

Culbertson et al. define overlapping clustering functors to map from a category of metric spaces and non-expansive mappings (such as **Met**, $\mathbf{Met}_{inj}$, or $\mathbf{Met}_{isom}$) to **Cov**. They then use this definition to define a generative model for representing non-overlapping clusterings in terms of finite metric spaces. Given a set of finite metric spaces $\mathscr{T}$, the clustering functor $\mathbf{ML}^{\mathscr{T}}$ maps the metric space (X, d_X) to the flag cover formed from the maximally linked subsets of the relation R such that xRx' if for some $(T, d_T) \in \mathscr{T}$ there exists a morphism (non-expansive map) $t : (T, d_T) \to (X, d_X)$ such that $x, x' \in Im(t)$.

The main difference between this construction and Carlsson and Mémoli (2013)'s generative model is the lack of transitivity. A clustering functor characterized by Carlsson and Memoli's construction

maps the points x, x' to the same partition if there exists a sequence of metric spaces in Ω whose images connect them. In contrast, a clustering functor characterized by Culbertson et al. (2016)'s construction will connect x and x' if there is a morphism from a single metric space in $\mathscr{T}$ that maps onto both of them.

To be specific, consider the clustering functor characterized by the collection $\mathscr{T} = \{\Delta_2(\delta)\}$ where $\Delta_2(\delta)$ is the finite 2-point metric space where the points are separated by distance δ. According to Carlsson and Mémoli (2013)'s generative model, this functor is the single linkage functor. However, according to Culbertson et al.'s generative model, this is instead the **maximal linkage at distance** δ functor, defined as follows:

Definition 4.8. *The maximal linkage at distance δ functor maps a metric space (X, d_X) to the set of maximally linked subsets of the relation R where $x_1 R x_2$ when $d(x_1, x_2) \leq \delta$.*

In Culbertson's model the single linkage functor is instead formed from the collection $\mathscr{T} = \{\Delta_0(\delta), \Delta_1(\delta), \Delta_2(\delta)), \Delta_3(\delta$ where $\Delta_n(\delta)$ is the finite n-point metric space where each pair of points is separated by distance δ.

In order to corral these differences Culbertson et al. use the language of simplicial complexes. Both single and maximal linkage can be defined in terms of the Vietoris-Rips complex. The clusters formed by single linkage are the connected components of the complex (which are non-overlapping) whereas the clusters formed by maximal linkage are the simplices of the complex (which may overlap along the faces).

In a follow-up paper, Culbertson et al. (2018) extend this construction to handle hierarchical clustering algorithms with overlaps. They first generalize Carlsson and Memoli's definition of persistent sets to **persistent covers** by replacing the partitions in the codomain with non-nested flag covers. Their construction is then built on top of functors from a category of metric space-like objects (which they call **weight categories**) to the category of **Sieves**, or persistent covers that contain the trivial cover $\{X\}$. Such a functor $\mathscr{C}$: **Weight** $\to$ **Sieve** is then a **stationary sieving functor** if:

- $\mathscr{C}$ commutes with the forgetful functor to **Set**.
- Given the functor $\mathscr{J}$: **Sieve** $\to$ **Weight** that maps the sieve (X, θ_X) to (X, d_X) where $d_X(x, x') = min\{t \mid \exists A \in \theta_X(t), x \in A, x' \in A\}$, the composition $\mathscr{C} \fatsemi \mathscr{J}$: **Weight** $\to$ **Weight** is idempotent and non-expansive.

Like Carlsson and Mémoli (2013), Culbertson et al. (2018) demonstrate that there exists a universal sieving functor through which every stationary sieving functor factors. However, unlike Carlsson and Memoli's construction, this functor is the maximal linkage functor (aka the **Rips Sieving** functor) rather than the single linkage functor. This highlights a fundamental difference between overlapping and non-overlapping clustering functors: maximal linkage is universal for overlapping clustering and single linkage is universal for non-overlapping clustering.

One thing that both Carlsson and Mémoli (2008, 2013) and Culbertson et al. (2016, 2018) highlight is the relationship between clusterings and simplicial complexes. For example, we can decompose Carlsson and Memoli's single linkage functor into the composition $F_\delta \fatsemi \pi_0$ where the functor F_δ : **Met** $\to$ **SCpx** maps a metric space to its δ-Rips Complex and π_0 : **SCpx** $\to$ **Set** is the connected components functor (Joyal and Tierney, 2008).

McInnes (2019) reframe hierarchical clustering from a topological perspective, and Shiebler (2020b) builds on this to explicitly characterize hierarchical overlapping clustering algorithms as functors from $(0, 1]^{\text{op}}$ to **Cov** that factor through a category of simplicial complexes. Shiebler also unites McInnes' perspective with that of Culbertson et al. (2016) by characterizing both the maximal and single linkage clustering functors in terms of a factorization through a finite singular set functor.

4.2.3 Flattening Hierarchical Clustering

One of the core theories of topological data analysis is that homological structures that exist at multiple scales are particularly important descriptors of a dataset (Chazal and Michel, 2017). For example, the important connected components of a filtration are those which have large differences between the indices of the simplicial complexes in which they first and last appear. McInnes and Healy (2017) and Chazal et al. (2013) use this insight to define strategies to flatten a hierarchical clustering that are akin to the derivation of a persistence diagram in topological data analysis (Chazal and Michel, 2017).

Rolle and Scoccola (2020) use Chazal et al. (2009)'s interleaving distance to explore the stability of these flattening algorithms.

Definition 4.9. *The **interleaving distance** between the functors $F, G : (\mathbb{R}, \leq) \to \mathscr{C}$ is the smallest ε such that there exist a pair of commuting natural transformations $G(r) \to F(r+\varepsilon)$ and $F(r) \to G(r+\varepsilon)$.*

If we cast an algorithm as a series of transformations between functors out of $(\mathbb{R}, \leq)$, we can prove that the algorithm is stable by proving that each transformation is uniformly continuous with respect to the interleaving distance (Scoccola, 2020).

4.2.4 Multiparameter Hierarchical Clustering

One of the shortcomings of the robust single linkage algorithm is that it requires an additional density estimation parameter, the choice of which can be arbitrary. Since this parameter behaves similarly to the scale parameter over which single linkage clustering varies, a natural question is whether hierarchical clustering can be extended to multiple parameters. This would both simplify the theoretical presentation of the algorithm and enable the algorithm to take advantage of the structure exposed by varying the scale parameter. Carlsson and Memoli explore this theme in "Multiparameter Hierarchical Clustering Methods" (Carlsson and Mémoli, 2010); defining a **multiparameter hierarchical clustering scheme** to be a functor from a category of metric spaces to the category of **persistent structures**, which are multiparameter extensions of persistent sets. They then extend the uniqueness theorem of single linkage clustering (Carlsson and Mémoli, 2008) to demonstrate that robust single linkage is the unique 2-dimensional hierarchical clustering functor that satisfies a 2-dimensional extension of the conditions in this theorem.

One challenge with using multiparameter hierarchical clustering algorithms is that the most common procedures for flattening hierarchical clusterings rely on the possibility of representing the clustering with a merge tree or dendrogram (Chazal et al., 2013; McInnes and Healy, 2017). When the hierarchical structure is indexed by a partial order like $(\mathbb{R}^n, \leq)$ rather than a total order like $(\mathbb{R}, \leq)$, this is no longer an option. It is possible that there are two clusters c in $H(a_1, a_2, ..., a_n)$ and c' in $H(a'_1, a'_2, ..., a'_n)$ such that c and c' are neither disjoint nor in a containment relationship. This is closely related to the phenomenon that unlike single parameter persistence modules, multiparameter persistence modules cannot be decomposed into a direct sum of simple modules (Lesnick and Wright, 2015).

In order to get around this limitation, some authors use adaptations of Lesnick and Wright (2015)'s strategy of using persistent structures that are parameterized along affine slices of $(\mathbb{R}^n, \leq)$. For example, Rolle and Scoccola (2020) define a hierarchical clustering algorithm that is indexed by a curve γ in $(\mathbb{R}^n, \leq)$, rather than all of $(\mathbb{R}^n, \leq)$. In this way the clustering represents structure across different values of each parameter. They use this strategy to define the γ-linkage algorithm, which is a multiparameter hierarchical generalization of robust single linkage.

Shiebler (2021a) takes a different perspective, and instead develops an algorithm based on binary integer programming and a prior distribution over hyperparameter values to solve the multiparameter

flattening problem directly. He demonstrates that this algorithm can produce better results than selecting an optimal hyperparameter value.

4.2.5 Functorial Manifold Learning

Some authors have also begun exploring functorial frameworks for manifold learning. McInnes et al. (2018) build a dimensionality reduction algorithm, UMAP (Uniform Manifold Approximation and Projection), that uses simplicial complexes and filtrations to define a coherent way to combine multiple local approximations of geodesic distance.

UMAP exploits the property that we can locally approximate the geodesic distance between points that lie on a manifold that is embedded within $\mathbb{R}^n$. In order to extend these local approximations into an approximation of the geodesic distance between any point x and its neighbors, UMAP defines a custom distance metric for x such that the data is uniformly distributed with respect to this metric. This metric defines the distance from x to any point y to be $\frac{1}{r} d_{\mathbb{R}^n}(x,y)$, where $d_{\mathbb{R}^n}(x,y)$ is the $\mathbb{R}^n$-distance from x to y and r is the distance from x to its nearest neighbor. This technique is similar to the strategies used in HDBSCAN and robust single linkage, but in the opposite direction. Rather than expand distances in regions of low density, UMAP contracts them. This effectively normalizes all regions to uniform density.

This rescaling creates a family of distinct metric spaces (one for each data point x). In order to combine these spaces into a coherent structure, UMAP builds a Rips complex for each metric space and then combines the complexes with a fuzzy set union. The resulting combined simplicial complex acts as a representation of the local and global connectivity of the dataset. This representation is used to learn an $n \times d$ matrix A of d-dimensional embeddings for the n elements of (X, d_X) by applying stochastic gradient descent to minimize the cross entropy between the reference simplicial complex and the Rips complex of the metric space defined by the embeddings in A.

Shiebler (2021b) builds on this to introduce a hierarchy of manifold learning algorithms based on the dataset transformations over which they are equivariant. Shiebler shows that UMAP is equivariant to isometries and both IsoMap (Tenenbaum et al., 2000) and Metric Multidimensional Scaling (Abdi, 2007) are equivariant to surjective non-expansive maps. Like Rolle and Scoccola (2020); Scoccola (2020), Shiebler also uses interleaving distance to bound on how well the embeddings that manifold learning algorithms learn on noisy data approximate the embeddings they learn on noiseless data.

4.3 Functorial Supervised Learning

Some authors have begun to use similar techniques to those described in Section 4.2 to characterize the invariances of supervised learning algorithms. Harris (2019) builds a framework in which learning algorithms are natural transformations between a training dataset functor D and a prediction model functor P. Given a category $\mathbf{X}$ of input spaces, a category $\mathbf{Y}$ of output spaces, and an index category $\mathbf{I}$:

- $D : \mathbf{X} \times \mathbf{Y} \times \mathbf{I}^{\text{op}} \to \mathbf{Set}$ maps an (input, output, index) tuple to the set of all possible training datasets $\{(x_i, y_i) \mid i \in I\}$.
- $P : \mathbf{X}^{op} \times \mathbf{Y} \times \mathbf{I}^{\text{op}} \to \mathbf{Set}$ maps an (input, output, index) tuple to the set of all possible prediction functions $f : \mathbf{X} \to \mathbf{Y}$.

Harris defines an invariant learning algorithm as a transformation $\mu : D \to P$ that is natural in $\mathbf{Y}$ and $\mathbf{I}$ and dinatural in $\mathbf{X}$. Intuitively such an algorithm defines a collection of functions, each of which maps a set of training datasets to a set of prediction functions. He then characterizes learning algorithms in terms of the categories $\mathbf{X}, \mathbf{Y}, \mathbf{I}$ over which they satisfy this naturality condition. For example, the linear

regression algorithm is natural when $\mathbf{X} = \mathbf{FinVec}_{iso}$ (the category of finite dimensional vector spaces and invertible linear maps), $\mathbf{Y} = \mathbf{FinVec}$ (the category of finite dimensional vector spaces and linear maps), and $\mathbf{I} = \mathbf{Euc}_{mono}$ (the category of Euclidean spaces and monomorphisms). However, linear regression is not natural when $\mathbf{X} = \mathbf{FinVec}$ because the normal equations require inverting the data matrix.

An another example, Healy (2000) model the relationship between "concepts", which they represent as theories in formal logic, and components of a cognitive neural network that learns these concepts. They define a category **Concpt** that has concepts as objects and subconcept relationship as morphisms and a category **Neural** with architectures as objects and sets of **priming states**, which represent how one part of the network can activate another, as morphisms. They define a functor $M : \mathbf{Concpt} \to \mathbf{Neural}$ to model how cognitive neural networks pass information about concepts, and they use colimits to formalize the construction of complex concepts from simpler ones.

Gavranović (2019) takes a slightly different perspective and focuses directly on the relationship between the neural network architecture and the associated optimization problem. He bases most of his constructions on the free category $\mathbf{Free}(G)$ of a graph G, which behaves somewhat like the neural network equivalent of a database schema (Spivak, 2012). One of the core concepts is the *Arch* functor, which maps arrows in $\mathbf{Free}(G)$ to parameterized functions, or arrows in the category **Para**. Since an arrow in $\mathbf{Free}(G)$ can be interpreted as a sequence of composed edges in a graph, *Arch* acts somewhat like a deep learning library: it transforms a connection template into a parameterized function.

In order to describe the process of resolving a parameterized function to a particular model (the training process) by partially applying a vector in R^p, Gavranović uses the following dependently typed function:

$$PSpec : (Arch : Ob(\mathbf{Para}^{\mathbf{Free}(G)})) \times \mathscr{P}(Arch) \to Ob(\mathbf{Euc}^{\mathbf{Free}(G)})$$

This function maps pairs of architectures and parameters to model functors $Model_p : Free(G) \to \mathbf{Euc}$. Although it seems as if there could exist some functor F that serves a similar role to *PSpec* but such that $Model_p$ factors into $Model_p = Arch \fatsemi F$, there are some problems with this. In order for this to work, it would need to be the case that for architectures a_1, a_2, the functor F maps $a_1 \fatsemi a_2$ to $Fa_1 \fatsemi Fa_2$, which is difficult to enforce. It is also very limiting: consider the networks $a_1 \fatsemi a_2$ and $a_1 \fatsemi a_3$. There is no reason to believe that the best weights for a_1 within $a_1 \fatsemi a_2$ are the same as the best weights for a_1 within $a_1 \fatsemi a_3$.

This problem sheds light on one of the main challenges with using categorical models of optimization and neural networks to reason about Machine Learning: relating an optimization system to the task it is trained with. A Machine Learning task is the optimization problem that a Machine Learning system is designed to solve. Typically this problem is defined in terms of an objective function over data samples that are assumed to be drawn from some probability distribution. There are Machine Learning tasks for which the best weights for a_1 within $a_1 \fatsemi a_2$ are the same as the best weights for a_1 within $a_1 \fatsemi a_3$, but specifying these tasks to be compatible with our specifications for Machine Learning systems is challenging.

Gavranović also takes a particularly abstract perspective on defining tasks. He uses the objects in $\mathbf{Free}(G)$ as templates for concepts (sets), such as "pictures of a horse". $Model_p$ then maps arrows in $\mathbf{Free}(G)$ to parameterized functions that transform between concepts. For example, a classification model will map the "pictures of a horse" concept to the Boolean "True" concept. Gavranović formalizes this notion with the **embedding** functor $E : |\mathbf{Free}(G)| \to \mathbf{Set}$, and describes a dataset D_E as a subfunctor of E that assigns a particular dataset to each concept.

4.4 Equivariant Neural Networks

A particularly important stream of machine learning research focuses on the equivariance and invariance properties of neural network architectures. Although much of this work has category theoretic overtones, many of the authors who contribute to this stream do not go through the trouble of defining categories and functors between them, and instead derive equivariance properties explicitly.

A function is equivariant to a transformation if applying that transformation to its inputs results in an equivalent transformation of its outputs. Invariance is a special case of equivariance in which any transformation of the inputs results in an identity transformation of the outputs. For example, the weight-sharing regimes of CNNs and RNNs make them equivariant to image translations and sequence position shifts respectively (up to edge effects).

For this reason, researchers are actively exploring neural network architectures with equivariance properies. Cohen and Welling (2016) generalize CNNs to G-CNNs, which use generalized convolution and pooling layers to exhibit equivariance to group transformations like rotations and reflections as well as translations. Intuitively, G-convolutions replace the position shifting in a discrete convolution operation with a general group of transformations. As group equivariance has grown in popularity, some authors have begun to dig deeper into its theoretical foundations. Kondor and Trivedi (2018) show that a neural network layer is G-equivariant only if it has convolution structure and Cohen et al. (2020) show that linear equivariant maps between the feature spaces are in one-to-one correspondence with convolutions using equivariant kernels.

Cohen et al. (2018) build on this work to develop a spherical CNN that uses generalized Fourier transformation-powered spherical convolutions to exhibit equivariance to sphere rotations. The authors demonstrate that their network is particularly effective at classifying 3D shapes that have been projected onto a sphere with ray casting.

Cohen et al. (2019) extend neural network equivariance beyond group symmetries. They develop a strategy to make neural networks that consume data on a general manifold dependent only on the intrinsic geometry of the manifold. In particular, the authors develop a convolution operator that is equivariant to the choice of **gauge**, or the tangent frame on the manifold relative to which the orientation of filters are defined. The authors implement gauge equivariant CNNs for a convenient type of manifold (the icosahedon) and demonstrate that this strategy yields state of the art performance at learning spherical signals.

One data type with particularly complex symmetry properties is graph data. Maron et al. (2019) develop linear operators that are equivariant to graph permutations and de Haan et al. (2020) make explicit use of category theory to build more powerful neural networks that can exploit graph symmetries. They start by introducing the concept of a graph representation, which is essentially a functor from the category of graphs and graph isomorphisms to vector spaces and linear maps. They then generalize Maron et al. (2019)'s equivariant graph networks to **global natural graph networks**. Each layer in a natural graph network is a natural transformation of graph representations. That is, it is a linear map that commutes with graph isomorphisms. This construction is both flexible and powerful: unlike a layer in an equivariant graph network (Maron et al., 2019) that applies the same equivariant transformation to each graph, a layer in a global natural graph networks can apply different transformations to different graphs as long as the transformations commute with the chosen graph representation.

A related stream of work by Bronstein et al. (2016) aims to generalize equivariant and invariant neural network operators to non-Euclidean domains (e.g. graphs and manifolds). This **geometric deep learning** perspective largely focuses on the characterization and generalization of the equivariance properties of convolution, and therefore naturally benefits from the universality of the convolution as an

invariant operator (Kondor and Trivedi, 2018) (Cohen et al., 2020). Bronstein et al. (2016) illustrate that convolution commutes with the Laplacian operator, and they use this insight to define both spatially and spectrally motivated generalizations of convolution for non-Euclidean domains.

5 Discussion

The role of category theory in machine learning research is still nascent, and we are excited to see how it grows over the next decade. The synthetic characterizations of gradient-based learning and probability theory are constantly developing, and the success of geometric deep learning (Bronstein et al., 2016) has drawn the mainstream machine learning community towards category theoretic ideas.

Looking forward, there are two areas in particular where we expect to see large strides. First, although some authors have begun to explore categorical generalizations of classical machine learning techniques like gradient descent and Bayesian updating (Cho and Jacobs, 2019; Wilson and Zanasi, 2021), there has been very little exploration of the convergence properties of these generalized algorithms. There is a need for a categorical perspective on learning theory to evolve along with the categorical perspective on machine learning. Second, categorical perspectives on machine learning have not yet been the unifying force that they have the potential to be: the synthetic perspectives on probability and gradient-based learning are largely disjoint, and are even farther removed from the research on equivariant and invariant learning. We hope to see more work focused on unification.

In the even longer term, understanding how optimal solutions – a concept at the heart of machine learning – can be understood through the lens of category theory is something that has not yet been studied in the literature. Optimal solutions to abstract problems are in category theory often characterized by their universal properties, and finding best approximations to problems is done by computing Kan extensions. We conjecture that applying these ideas to concrete problems could be fruitful in providing a deep, overarching foundation to machine learning and optimization theory.

Finally, for the most part we have avoided discussing the more application-focused intersections between category theory and machine learning in this work. This includes categorical perspectives on natural language processing, automata learning, quantum machine learning, and many other areas. These subfields are developing extremely quickly and we look forward to future surveys which cover them.

References

Context and compositionality in biological and artificial neural systems (NeurIPS workshop). 2019. URL `https://context-composition.github.io/`.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Hervé Abdi. Metric multidimensional scaling (MDS): analyzing distance matrices. *Encyclopedia of Measurement and Statistics*, 2007.

Lloyd Allison. Types and classes of machine learning and data mining. In *Proceedings of the 26th Australasian computer science conference-Volume 16*, pages 207–215. Australian Computer Society, Inc., 2003.

François Bancilhon and Nicolas Spyratos. Update semantics of relational views. *ACM Transactions on Database Systems (TODS)*, 6(4):557–575, 1981.

Aaron Bohannon, J. Nathan Foster, Benjamin C. Pierce, Alexandre Pilkiewicz, and Alan Schmitt. Boomerang: Resourceful lenses for string data. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '08, page 407–419, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781595936899. doi: 10.1145/1328438.1328487. URL https://doi.org/10.1145/1328438.1328487.

Tai-Danae Bradley. What is applied category theory? *arXiv e-prints arXiv:1809.05923*, 2018.

Denny Britz. Ai research, replicability, and incentives. 2020. https://dennybritz.com/blog/ai-replication-incentives/.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *CoRR*, abs/1611.08097, 2016. URL http://arxiv.org/abs/1611.08097.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020. URL https://arxiv.org/abs/2005.14165.

Joe Brucker. Category theory and natural language processing. 2020. URL https://github.com/jbrkr/Category_Theory_Natural_Language_Processing_NLP.

R. M. Burstall and John Darlington. A transformation system for developing recursive programs. *J. ACM*, 24(1):44–67, January 1977. ISSN 0004-5411. doi: 10.1145/321992.321996. URL https://doi.org/10.1145/321992.321996.

Gunnar Carlsson and Facundo Mémoli. Persistent clustering and a theorem of J. Kleinberg. *arXiv e-prints arXiv:0808.2241*, 2008.

Gunnar Carlsson and Facundo Mémoli. Multiparameter hierarchical clustering methods. In *Classification as a Tool for Research*, pages 63–70. Springer, 2010.

Gunnar Carlsson and Facundo Mémoli. Classifying clustering schemes. *Foundations of Computational Mathematics*, 13(2):221–252, 2013.

Kamalika Chaudhuri and Sanjoy Dasgupta. Rates of convergence for the cluster tree. In *Advances in neural information processing systems*, pages 343–351, 2010.

Frédéric Chazal and Bertrand Michel. An introduction to topological data analysis: fundamental and practical aspects for data scientists. *arXiv e-prints arXiv:1710.04019*, 2017.

Frédéric Chazal, David Cohen-Steiner, Marc Glisse, Leonidas J Guibas, and Steve Y Oudot. Proximity of persistence modules and their diagrams. In *Proceedings of the twenty-fifth annual symposium on Computational geometry*, pages 237–246, 2009.

Frédéric Chazal, Leonidas J Guibas, Steve Y Oudot, and Primoz Skraba. Persistence-based clustering in Riemannian manifolds. *Journal of the ACM (JACM)*, 60(6):1–38, 2013.

Kenta Cho and Bart Jacobs. Disintegration and Bayesian inversion via string diagrams. *Mathematical Structures in Computer Science*, 29(7):938–971, 2019.

Bryce Clarke, Derek Elkins, Jeremy Gibbons, Fosco Loregian, Bartosz Milewski, Emily Pillmore, and Mario Román. Profunctor optics, a categorical update, 2020.

Robin Cockett, Geoffrey Cruttwell, Jonathan Gallagher, Jean-Simon Pacaud Lemay, Benjamin MacAdam, Gordon Plotkin, and Dorette Pronk. Reverse derivative categories. *arXiv e-prints arXiv:1910.07065*, 2019.

Bob Coecke and Robert W. Spekkens. Picturing classical and quantum Bayesian inference. *arXiv e-prints*, art. arXiv:1102.2368, February 2011.

Taco Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. *arXiv e-prints arXiv:1811.02017*, 2020.

Taco S. Cohen and Max Welling. Group equivariant convolutional networks. *arXiv e-prints arXiv:1602.07576*, 2016.

Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical cnns. *arXiv e-prints arXiv:1801.10130*, 2018.

Taco S. Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. *arXiv e-prints arXiv:1902.04615*, 2019.

David Corfield. Collection of references on probability. 2021. URL `https://ncatlab.org/davidcorfield/show/probability#references`.

G. S. H. Cruttwell, Bruno Gavranović, Neil Ghani, Paul Wilson, and Fabio Zanasi. Categorical foundations of gradient-based learning. *arXiv e-prints arXiv:2103.01931*, 2021.

Jared Culbertson and Kirk Sturtz. Bayesian machine learning via category theory. *arXiv e-prints arXiv:1312.1445*, 2013.

Jared Culbertson and Kirk Sturtz. A categorical foundation for Bayesian probability. *Applied Categorical Structures*, 22(4):647–662, 2014.

Jared Culbertson, Dan P Guralnik, Jakob Hansen, and Peter F Stiller. Consistency constraints for overlapping data clustering. *arXiv e-prints arXiv:1608.04331*, 2016.

Jared Culbertson, Dan P Guralnik, and Peter F Stiller. Functorial hierarchical clustering with overlaps. *Discrete Applied Mathematics*, 236:108–123, 2018.

Pim de Haan, Taco Cohen, and Max Welling. Natural graph networks. *arXiv e-prints arXiv:2007.08349*, 2020.

Ernst-Erich Doberkat. Characterizing the eilenberg-moore algebras for a monad of stochastic relations. 2004.

Conal Elliott. The simple essence of automatic differentiation. *Proceedings of the ACM on Programming Languages*, 2(ICFP):1–29, 2018.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining (KDD)*, volume 96, pages 226–231, 1996.

Arnold M Faden et al. The existence of regular conditional probabilities: necessary and sufficient conditions. *The Annals of Probability*, 13(1):288–298, 1985.

Brendan Fong. Causal theories: A categorical perspective on Bayesian networks. *arXiv e-prints arXiv:1301.6201*, 2013.

Brendan Fong and Michael Johnson. Lenses and learners. *arXiv e-prints arXiv:1903.03671*, 2019.

Brendan Fong and David I Spivak. Seven sketches in compositionality: An invitation to applied category theory, 2018.

Brendan Fong, David Spivak, and Rémy Tuyéras. Backprop as functor: A compositional perspective on supervised learning. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–13. IEEE, 2019.

Uwe Franz. What is stochastic independence? In *Non-commutativity, infinite-dimensionality and probability at the crossroads*, pages 254–274. World Scientific, 2002.

Tobias Fritz. Convex Spaces I: Definition and Examples. *arXiv e-prints*, art. arXiv:0903.5522, March 2009.

Tobias Fritz. A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020.

Tobias Fritz, Tomáš Gonda, Paolo Perrone, and Eigil Fjeldgren Rischel. Representable Markov Categories and Comparison of Statistical Experiments in Categorical Probability. *arXiv e-prints*, art. arXiv:2010.07416, October 2020.

Bruno Gavranovic. Compositional deep learning. *arXiv e-prints arXiv:1907.08292*, 2019.

Bruno Gavranović. Learning functors using gradient descent. *Applied Category Theory*, 2019.

Neil Ghani, Jules Hedges, Viktor Winschel, and Philipp Zahn. A compositional approach to economic game theory. *arXiv e-prints arXiv:1603.04641*, 2016.

Michele Giry. A categorical approach to probability theory. In *Categorical aspects of topology and analysis*, pages 68–85. Springer, 1982.

William H Guss and Ruslan Salakhutdinov. On universal approximation by neural networks with uniform guarantees on approximation of infinite dimensional maps. *arXiv e-prints arXiv:1910.01545*, 2019.

Kenneth D. Harris. Characterizing the invariances of learning algorithms using category theory. *arXiv e-prints arXiv:1905.02072*, 2019.

Michael J Healy. Category theory applied to neural modeling and graphical representations. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pages 35–40. IEEE, 2000.

Jules Hedges. Limits of bimorphic lenses. *arXiv e-prints*, art. arXiv:1808.05545, August 2018.

Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. A convenient category for higher-order probability theory. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2017.

Chris Heunen, Ohad Kammar, Sam Staton, Sean Moss, Matthijs Vákár, Adam Ścibior, and Hongseok Yang. The semantic structure of quasi-borel spaces. *Probabilistic Programming Languages, Semantics, and Systems*, 2018.

Alex Irpan. Deep reinforcement learning doesn't work yet. https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

Bart Jacobs. A channel-based perspective on conjugate priors. *CoRR*, abs/1707.00269, 2017.

Bart Jacobs. Categorical aspects of parameter learning. *arXiv e-prints arXiv:1810.05814*, 2018.

Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *arXiv e-prints arXiv:2101.11174*, 2021.

André Joyal and Myles Tierney. Notes on simplicial homotopy theory. *preprint*, 2008. URL https://ncatlab.org/nlab/files/JoyalTierneyNotesOnSimplicialHomotopyTheory.pdf.

Michael J Kearns, Robert E Schapire, and Linda M Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

Andrew Kennedy. Compiling with continuations, continued. In *Proceedings of the 12th ACM SIGPLAN international conference on Functional programming*, pages 177–190, 2007.

Jon M Kleinberg. An impossibility theorem for clustering. In *Advances in neural information processing systems*, pages 463–470, 2003.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *arXiv e-prints arXiv:1802.03690*, 2018.

F William Lawvere. The category of probabilistic mappings. *preprint*, 1962. URL https://ncatlab.org/nlab/files/lawvereprobability1962.pdf.

Michael Lesnick and Matthew Wright. Interactive visualization of 2-d persistence modules. *arXiv e-prints arXiv:1512.00180*, 2015.

Drew Linsley, Dan Shiebler, Sven Eberhardt, and Thomas Serre. Learning what and where to attend with humans in the loop. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJgLg3R9KQ.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv e-prints arXiv:1812.09902*, 2019.

Peter McCullagh. What is a statistical model? *Annals of statistics*, pages 1225–1267, 2002.

Leland McInnes. Topological methods for unsupervised learning. In *International Conference on Geometric Science of Information*, pages 343–350. Springer, 2019.

Leland McInnes and John Healy. Accelerated hierarchical density clustering. *arXiv e-prints arXiv:1705.07321*, 2017.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv e-prints arXiv:1802.03426*, 2018.

Chris Olah and Shan Carter. Research debt. *Distill*, 2017. doi: 10.23915/distill.00005. https://distill.pub/2017/research-debt.

Christopher Olah. Neural networks, types, and functional programming. 2015. https://colah.github.io/posts/2015-09-NN-Types-FP/.

Ali Rahimi. Machine learning has become alchemy. 2018. https://www.youtube.com/watch?v=x7psGHgatGM.

Bastian Rieck. Machine learning needs a langlands programme. 2020. https://bastian.rieck.me/blog/posts/2020/langlands/.

Mitchell Riley. Categories of optics. *arXiv e-prints arXiv:1809.00738*, 2018.

Alexander Rolle and Luis Scoccola. Stable and consistent density-based clustering. *arXiv e-prints arXiv:2005.09048*, 2020.

Mario Román. Open diagrams via coend calculus. *Electronic Proceedings in Theoretical Computer Science*, 333:65–78, Feb 2021. ISSN 2075-2180. doi: 10.4204/eptcs.333.5. URL http://dx.doi.org/10.4204/EPTCS.333.5.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv e-prints arXiv:1609.04747*, 2017.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL https://www.nature.com/articles/323533a0. Number: 6088 Publisher: Nature Publishing Group.

Venu Satuluri, Yao Wu, Xun Zheng, Yilei Qian, Brian Wichers, Qieyun Dai, Gui Ming Tang, Jerry Jiang, and Jimmy Lin. Simclusters: Community-based representations for heterogeneous recommendations at twitter. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 3183–3193, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3403370. URL https://doi.org/10.1145/3394486.3403370.

Luis N Scoccola. Locally persistent categories and metric properties of interleaving distances. *Thesis, University of Western Ontario*, 2020. URL https://ir.lib.uwo.ca/cgi/viewcontent.cgi?article=9630&context=etd.

Robert A.G. Seely, Richard F. Blute, and J. R. B Cockett. Differential categories. *Mathematical structures in computer science*, 16(6):1049–1083, 2006.

Robert A.G. Seely, Richard F. Blute, and J. R. B Cockett. Cartesian differential categories. *Theory and Applications of Categories*, 22(23):622–672, 2009.

Dan Shiebler. Categorical stochastic processes and likelihood. *arXiv e-prints arXiv:2005.04735*, 2020a.

Dan Shiebler. Functorial clustering via simplicial complexes. *Topological Data Analysis and Beyond Workshop at NeurIPS 2020*, 2020b. URL https://openreview.net/pdf?id=ZkDLcXCP5sV.

Dan Shiebler. Flattening multiparameter hierarchical clustering functors. *International Conference on Geometric Science of Information*, 2021a. URL https://arxiv.org/pdf/2104.14734.pdf.

Dan Shiebler. Functorial manifold learning. *arXiv e-prints arXiv:2011.07435*, 2021b.

Dan Shiebler, Luca Belli, Jay Baxter, Hanchen Xiong, and Abhishek Tayal. Fighting redundancy and model decay with embeddings. *CoRR*, abs/1809.07703, 2018. URL http://arxiv.org/abs/1809.07703.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017. ISSN 1476-4687. doi: 10.1038/nature24270. URL https://www.nature.com/articles/nature24270/. Number: 7676 Publisher: Nature Publishing Group.

Alex Simpson. Category-theoretic structure for independence and conditional independence. In Sam Staton, editor, *Proceedings of the Thirty-Fourth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2018, Dalhousie University, Halifax, Canada, June 6-9, 2018*, volume 341 of *Electronic Notes in Theoretical Computer Science*, pages 281–297. Elsevier, 2018. doi: 10.1016/j.entcs.2018.03.028. URL https://doi.org/10.1016/j.entcs.2018.03.028.

Toby St. Clere Smithe. Bayesian updates compose optically. *arXiv e-prints arXiv:2006.01631*, 2020.

David I Spivak. Functorial data migration. *Information and Computation*, 217:31–51, 2012.

David I. Spivak. Generalized lens categories via functors $\mathscr{C}^{\mathrm{op}} \to \mathsf{Cat}$. *arXiv e-prints arXiv:1908.02202*, 2020a.

David I. Spivak. Poly: An abundant categorical setting for mode-dependent dynamics. *arXiv e-prints arXiv:2005.01894*, 2020b.

David Sprunger and Shin-ya Katsumata. Differentiable causal computations via delayed trace. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12. IEEE, 2019.

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

Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity*, pages 11–30. Springer, 2015.

Paul Wilson and Fabio Zanasi. Reverse derivative ascent: A categorical approach to learning boolean circuits. *Electronic Proceedings in Theoretical Computer Science*, 333:247–260, Feb 2021. ISSN 2075-2180. doi: 10.4204/eptcs.333.17. URL http://dx.doi.org/10.4204/EPTCS.333.17.

David Wishart. 256. note: An algorithm for hierarchical classifications. *Biometrics*, pages 165–170, 1969.