Machine Learning and Invariant Theory

Ben Blum-Smith and Soledad Villar

1. Introduction

Modern machine learning has not only surpassed the state of the art in many engineering and scientific problems, but it also has had an impact on society at large, and will likely continue to do so. This includes deep learning, large language models, diffusion models, etc. In this article, we give an account of certain mathematical principles that are used in the definition of some of these machine learning models, and we explain how classical invariant theory plays a role in them. Due to space constraints we leave out many relevant references. A version of this manuscript with a longer set of references is available on arXiv [1].

In supervised machine learning, we typically have a training set $(x_i, y_i)_{i=1}^n$, where $x_i \in \mathbb{R}^d$ are the data points and $y_i \in \mathbb{R}^k$ are the labels. A typical example is image recognition, where the x_i are images and the y_i are image labels (say, "cat" or "dog"), encoded as vectors. The goal is to find a function $\hat{f}$ in a hypothesis space $\mathcal{F}$, that not only approximately interpolates the training data ($\hat{f}(x_i) \approx y_i$), but also performs well on unseen (or held-out) data. The function $\hat{f}$ is called the *trained model*, *predictor*, or *estimator*. In practice, one parametrizes the class of functions $\mathcal{F}$ with some parameters θ varying over a space Θ of parameter values sitting inside some $\mathbb{R}^s$; in other words, $\mathcal{F} = \{f_\theta : \mathbb{R}^d \to \mathbb{R}^k, \theta \in \Theta \subseteq \mathbb{R}^s\}$. Then one uses local optimization (in θ) to find a function in $\mathcal{F}$ that locally and approximately minimizes a prespecified empirical loss function ℓ which compares a candidate function f_θ's values on the

Ben Blum-Smith is a postdoctoral fellow in the department of applied mathematics & statistics at the Johns Hopkins University. His email address is bblumsm1@jhu.edu.

Soledad Villar is an assistant professor in the department of applied mathematics & statistics at the Johns Hopkins University. Her email address is svillar3@jhu.edu.

For permission to reprint this article, please contact: reprint-permission@ams.org.

DOI: https://doi.org/10.1090/noti2760

x_i with the "true" target values y_i. In other words, one approximately solves $\theta^* := \operatorname{argmin}_\theta \sum_{i=1}^n \ell(f_\theta(x_i), y_i)$, and then takes $\hat{f} = f_{\theta^*}$.

Modern machine learning performs regressions on classes of functions that are typically overparameterized (the dimension s of the space of parameters is much larger than the number n of training samples), and in many cases, several functions in the hypothesis class $\mathcal{F}$ can interpolate the data perfectly. Deep learning models can even interpolate plain noise, or fit images to random labels. Moreover, the optimization problem is typically nonconvex. Therefore the model performance is highly dependent on how the class of functions is parameterized and the optimization algorithms employed.

The parameterization of the hypothesis class of functions is what in deep learning is typically referred to as *the architecture*. In recent years, the most successful architectures have been ones that use properties or heuristics regarding the structure of the data (and the problem) to design the class of functions: convolutional neural networks for images, recurrent neural networks for time series, graph neural networks for graph-structured data, transformers, etc. Many of these design choices are related to the symmetries of the problem: for instance, convolutional neural networks can be translation equivariant, and transformers can be permutation invariant.

When the learning problem comes from the physical sciences, there are concrete sets of rules that the function being modeled must obey, and these rules often entail symmetries. The rules (and symmetries) typically come from coordinate freedoms and conservation laws [19]. One classical example of these coordinate freedoms is the scaling symmetry that comes from dimensional analysis (for instance, if the input data to the model is rescaled to change everything that has units of kilograms to pounds, the predictions should scale accordingly). In order to do machine learning on physical systems, researchers have designed models that are consistent with physical law; this is the case for physics-informed machine learning, neural ODEs and PDEs, and equivariant machine learning.

Given data spaces V, W and a group G acting on both of them, a function $f : V \to W$ is equivariant if $f(g \cdot v) = g \cdot f(v)$ for all $g \in G$ and all $v \in V$. Many physical problems are equivariant with respect to rotations, permutations, or scalings. For instance, consider a problem where one uses data to predict the dynamics of a folding protein or uses simulated data to emulate the dynamics of a turbulent fluid. Equivariant machine learning restricts the hypothesis space to a class of equivariant functions. The philosophy is that every function that the machine learning model can express is equivariant, and therefore consistent with physical law.

Symmetries were used for machine learning (and in particular neural networks) in early works [16], and more recently they have been revisited in the context of deep learning. There are three main ways to implement symmetries. The simplest one parameterizes the invariant and equivariant functions with respect to discrete groups by averaging arbitrary functions over the group orbit [3]. The second approach, explained in the next section, uses classical representation theory to parameterize the space of equivariant functions (see for instance [8]). The third approach, the main point of this article, uses invariant theory.

As an example, we briefly discuss graph neural networks (GNNs), which have been a very popular area of research in the past couple of years. GNNs can be seen as equivariant functions that take a graph represented by its adjacency matrix $A \in \mathbb{R}^{n\times n}$ and possible node features $X \in \mathbb{R}^{n\times d}$, and output an embedding $f(A, X) \in \mathbb{R}^{n\times d}$ so that $f(\Pi A \Pi^\top, \Pi X) = \Pi f(A, X)$ for all Π $n \times n$ permutation matrices. Graph neural networks are typically implemented as variants of graph convolutions or message passing, which are equivariant by definition. However, many equivariant functions cannot be expressed with these architectures. Several recent works analyze the expressive power of different GNN architectures in connection to the graph isomorphism problem.

Beyond graphs, equivariant machine learning models have been extremely successful at predicting molecular structures and dynamics, protein folding, protein binding, and simulating turbulence and climate effects, to name a few applications. Theoretical developments have shown the universality of certain equivariant models, as well as generalization improvements of equivariant machine learning models over nonequivariant baselines. There has been some recent work studying the inductive bias of equivariant machine learning, and its relationship with data augmentation. See [1] for a list of references on these topics.

2. Equivariant Convolutions and Multilayer Perceptrons

Modern deep learning models have evolved from the classical artificial neural network known as the perceptron. The multilayer perceptron model takes an input x and outputs $F(x)$ defined to be the composition of affine linear maps and nonlinear entry-wise functions. Namely,

$$F(x) = \rho \circ L_T \circ \ldots . \circ L_2 \circ \rho \circ L_1(x)\,, \tag{1}$$

where ρ is the (fixed) entrywise nonlinear function and $L_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_{i+1}}$ are affine linear maps to be learned from the data. The linear maps L_i can be expressed as $L_i(x) = A_i x + b_i$ where $A_i \in \mathbb{R}^{d_i \times d_{i+1}}$ and $b_i \in \mathbb{R}^{d_{i+1}}$. In this example each function F is defined by the parameters $\theta = (A_i, b_i)_{i=1}^T$.

The first neural network that was explicitly equivariant with respect to a group action is the convolutional neural network. The observation is that if the $N \times N$ input images x are seen in the torus $\mathbb{Z}/N\mathbb{Z} \times \mathbb{Z}/N\mathbb{Z}$, the linear equivariant maps are cross-correlations (which in ML are referred to as convolutions) with fixed filters. The idea of restricting the linear maps to satisfy symmetry constraints was generalized to equivariance with respect to discrete rotations and translations, and to general homogenous spaces. Note that when working with arbitrary groups there are restrictions on the functions ρ for the model to be equivariant.

Classical results in neural networks show that certain multilayer perceptrons can universally approximate any continuous function in the limit where the number of neurons goes to infinity. However, that is not true in general in the equivariant case. Namely, functions expressed as (1) where the L_i are linear and equivariant may not universally approximate all continuous equivariant functions. In some cases, there may not even exist nontrivial linear equivariant maps.

One popular idea to address this issue is to extend this model to use equivariant linear maps on tensors. Now $L_i : \mathbb{R}^{d^{\otimes k_i}} \to \mathbb{R}^{d^{\otimes k_{i+1}}}$ are linear equivariant maps (where the action in the tensor product is defined as the tensor product of the action in each component and extended linearly). Now the question is how can we parameterize the space of such functions to do machine learning? The answer is via Schur's lemma.

A representation of a group G is a map $\phi : G \to \mathrm{GL}(V)$ that satisfies $\phi(g_1 g_2) = \phi(g_1)\phi(g_2)$ (where V is a vector space and $\mathrm{GL}(V)$, as usual, denotes the automorphisms of V, that is, invertible linear maps $V \to V$). A group action of G on $\mathbb{R}^d$ (written as $\cdot$) is equivalent to the group representation $\phi : G \to \mathrm{GL}(\mathbb{R}^d)$ such that $\phi(g)(v) = g \cdot v$. We extend the action $\cdot$ to the tensor product $(\mathbb{R}^d)^{\otimes k}$ so that the group acts independently in every tensor factor (i.e., in every dimension or mode), namely $\phi_k = \otimes_{r=1}^k \phi : G \to \mathrm{GL}((\mathbb{R}^d)^{\otimes k})$.

The first step is to note that a linear equivariant map $L_i : (\mathbb{R}^d)^{\otimes k_i} \to (\mathbb{R}^d)^{\otimes k_{i+1}}$ corresponds to a map between group representations such that $L_i \circ \phi_{k_i}(g) = \phi_{k_{i+1}}(g) \circ L_i$ for all $g \in G$. Homomorphisms between group representations are easily parametrizable if we decompose the representations in terms of irreducible representations (aka irreps):

$$\phi_{k_i} = \bigoplus_{\ell=1}^{T_{k_i}} \mathcal{T}_\ell \, . \qquad (2)$$

In particular, Schur's Lemma says that a map between two irreps over $\mathbb{C}$ is zero (if they are not isomorphic) or a multiple of the identity (if they are).

The equivariant neural-network approach consists in decomposing the group representations in terms of irreps and explicitly parameterizing the maps [9]. In general, it is not obvious how to decompose an arbitrary group representation into irreps. However in the case where $G = \mathrm{SO}(3)$, the decomposition of a tensor representation as a sum of irreps is given by the Clebsch–Gordan decomposition:

$$\otimes_{s=1}^k \phi_s = \oplus_{\ell=1}^T \mathcal{T}_\ell \qquad (3)$$

The Clebsch–Gordan decomposition not only gives the decomposition of the right side of (3) but also it gives the explicit change of coordinates. This decomposition is fundamental for implementing the equivariant 3D point-cloud methods defined in [8] and other works (see references in [1]). Moreover, recent work [6] shows that the classes of functions used in practice are universal, meaning that every continuous SO(3)-equivariant function can be approximated uniformly in compact sets by those neural networks. However, there exists a clear limitation to this approach: Even though decompositions into irreps are broadly studied in mathematics (plethysm), the explicit transformation that allows us to write the decomposition of tensor representations into irreps is a hard problem in general. It is called the *Clebsch–Gordan problem*.

3. Invariant Theory for Machine Learning

An alternative but related approach to the linear equivariant layers described above is the approach based on invariant theory, the focus of this article. In particular, the authors of this note and collaborators [18] explain that for some physically relevant groups—the orthogonal group, the special orthogonal group, and the Lorentz group—one can use classical invariant theory to design universally expressive equivariant machine learning models that are expressed in terms of the generators of the algebra of invariant polynomials. Following an idea attributed to B. Malgrange (that we learned from G. Schwarz), it is shown how to use the generators of the algebra of invariant polynomials to produce a parameterization of equivariant functions for a specific set of groups and actions.

To illustrate, let us focus on $\mathrm{O}(d)$-equivariant functions, namely functions $f : (\mathbb{R}^d)^n \to \mathbb{R}^d$ such that $f(Qv_1, \dots, Qv_n) = Qf(v_1, \dots, v_n)$ for all $Q \in \mathrm{O}(d)$ and all $v_1, \dots, v_n \in \mathbb{R}^d$ (for instance, the prediction of the position and velocity of the center of mass of a particle system). The method of B. Malgrange (explicated below) leads to the conclusion that all such functions can be expressed as

$$f(v_1, \dots, v_n) = \sum_{j=1}^n f_j(v_1, \dots, v_n) v_j, \qquad (4)$$

where $f_j : (\mathbb{R}^d)^n \to \mathbb{R}$ are $\mathrm{O}(d)$-invariant functions. Classical invariant theory shows that f_j is $\mathrm{O}(d)$-invariant if and only if it is a function of the pairwise inner products

$(v_i^\top v_j)_{i,j=1}^n$. So, in actuality, (4) can be rewritten

$$f(v_1, \dots, v_n) = \sum_{j=1}^n p_j\left((v_i^\top v_j)_{i,j=1}^n\right) v_j, \tag{5}$$

where $p_j : \mathbb{R}^{n+\binom{n}{2}} \to \mathbb{R}$ are arbitrary functions. In other words, the pairwise inner products generate the algebra of invariant polynomials for this action, and every equivariant map is a linear combination of the vectors themselves with coefficients in this algebra.

In this article, we explicate the method of B. Malgrange in full generality, showing how to convert knowledge of the algebra of invariant polynomials into a characterization of the equivariant polynomial (or smooth) maps. In Section 4 we explain the general philosophy of the method, and in Section 5 we give the precise algebraic development, formulated as an algorithm to produce parametrizations of equivariant maps given adequate knowledge of the underlying invariant theory. In Section 6, we work through several examples.

We note that, for machine learning purposes, it is not critical that the functions p_j are defined on invariant *polynomials*, nor that they themselves are polynomials. In the following, we focus on polynomials because the ideas were developed in the context of invariant theory; the arguments explicated below are set in this classical context. However, in [18], the idea is to preprocess the data by converting the tuple $(v_j)_{j=1}^n$ to the tuple of dot products $(v_i^\top v_j)_{i,j=1}^n$, and then treat the latter as input to the p_j's, which are then learned using a machine learning architecture of one's choice. Therefore, the p_j's are not polynomials but belong to whatever function class is output by the chosen architecture. Meanwhile, some recent works [2, 5, 11] have proposed alternative classes of separating invariants that can be used in place of the classical algebra generators as input to the p_j's, and may have better numerical stability properties. This is a promising research direction.

4. Big Picture

We are given a group G, and finite-dimensional linear G-representations V and W over a field k. (We can take $k = \mathbb{R}$ or $\mathbb{C}$.) We want to understand the equivariant polynomial maps $V \to W$. We assume we have a way to understand G-*invariant* polynomials on spaces related to V and W, and the goal is to leverage that knowledge to understand the equivariant maps.

The following is a philosophical discussion, essentially to answer the question: why should it be possible to do this? It is not precise; its purpose is just to guide thinking. Below in Section 5 we show how to actually compute the equivariant polynomials $V \to W$ given adequate knowledge of the invariants. That section is rigorous.

The first observation is that any reasonable family of maps $V \to W$ (for example linear, polynomial, smooth, continuous, etc.) has a natural G-action induced from the actions on V and W, and that the G-equivariant maps in such a family are precisely the fixed points of this action, as we now explain. This observation is a standard insight in representation and invariant theory.

Let $\mathrm{Maps}(V, W)$ be the set of maps of whatever kind, and let $GL(V)$ (respectively $GL(W)$) be the group of linear invertible maps from V (respectively W) to itself. Given $f \in \mathrm{Maps}(V, W)$ and $g \in G$ and $v \in V$, we define the map gf by

$$gf \coloneqq \psi(g) \circ f \circ \phi(g^{-1}), \tag{6}$$

where $\phi : G \to GL(V)$ and $\psi : G \to GL(W)$ are the group homomorphisms defining the representations V and W. The algebraic manipulation to verify that this is really a group action is routine and not that illuminating. A perhaps more transparent way to understand this definition of the action as "the right one" is that it is precisely the formula needed to make this square commute:

$$\begin{tikzcd} V \arrow[r, "f"] \arrow[d, "\phi(g)"] & W \arrow[d, "\psi(g)"] \\ V \arrow[r, "gf"] & W \end{tikzcd}$$

It follows from the definition of this action that the condition $gf = f$ is equivalent to the statement that f is G-equivariant. The square above automatically commutes, so $gf = f$ is the same as saying that the below square commutes—

$$\begin{tikzcd} V \arrow[r, "f"] \arrow[d, "\phi(g)"] & W \arrow[d, "\psi(g)"] \\ V \arrow[r, "f"] & W \end{tikzcd}$$

—and this is what it means to be equivariant.

An important special case of (6) is the action of G on V^*, the linear dual of V. This is the case $W = k$ with trivial action, and for $\ell \in V^*$ (6) reduces to $g\ell \coloneqq \ell \circ \phi(g^{-1})$. This is known as the *contragredient action*. We will utilize it momentarily with W in the place of V.

The second observation is that $\mathrm{Maps}(V, W)$ can be identified with functions from a bigger space to the underlying field k by "currying," and this change in point of view preserves the group action. Again, this is a standard maneuver in algebra. Specifically, given any map $f \in \mathrm{Maps}(V, W)$, we obtain a function $\tilde{f} : V \times W^* \to k$, defined by the formula

$$\begin{aligned} \tilde{f} : V \times W^* &\to k \\ (v, \ell) &\mapsto \ell(f(v)). \end{aligned}$$

Note that the function $\tilde{f}$ is linear homogeneous in $\ell \in W^*$. Conversely, given any function $f' : V \times W^* \to k$ that is linear homogeneous in the second coordinate, we can recover a map $f : V \to W$ such that $f' = \tilde{f}$, by taking $f(v)$ to be the element of W identified along the canonical isomorphism $W \to W^{**}$ with the functional on W^* that sends $\ell \in W^*$ to $f'(v, \ell)$—this functional is guaranteed to exist by the fact that f' is linear homogeneous in the second coordinate. An observation we will exploit in the next section is that the desired functional is actually the gradient of $f'(v, \ell)$ with respect to ℓ.

This construction gives an identification of $\mathrm{Maps}(V, W)$ with a subset of $\mathrm{Maps}(V \times W^*, k)$. Furthermore, there is a natural action of G on $\mathrm{Maps}(V \times W^*, k)$, defined precisely by the above formula (6) with $V \times W^*$ in place of V, k in place of W, and trivial action on k;[1] and the identification described here preserves this action. Therefore, the fixed points for the G-action on $\mathrm{Maps}(V, W)$ correspond with fixed points for the G-action on $\mathrm{Maps}(V \times W^*, k)$, which are *invariant functions* (since the action of G on k is trivial).

What has been achieved is the reinterpretation of equivariant maps $f \in \mathrm{Maps}(V, W)$ first as fixed points of a G-action, and then as invariant functions $\tilde{f} \in \mathrm{Maps}(V \times W^*, k)$. Thus, knowledge of invariant functions can be parlayed into knowledge of equivariant maps.

5. Equivariance from Invariants

With the above imprecise philosophical discussion as a guide, Algorithm 1 shows how in practice to get from a description of invariant polynomials on $V \times W^*$, to equivariant polynomial (or smooth) maps $V \to W$. The technique given here is attributed to B. Malgrange; see [13, Proposition 2.2] where it is used to obtain the smooth equivariant maps, and [15, Proposition 6.8] where it is used to obtain holomorphic equivariant maps. Variants on this method are used to compute equivariant maps in [7, Sections 2.1–2.2], [12, Section 3.12], [4, Section 4.2.3], and [20, Section 4].

The goal of the algorithm is to provide a parametrization of equivariant maps. That said, the proof of correctness is constructive: as an ancillary benefit, it furnishes a method for taking an arbitrary equivariant map given by explicit polynomial expressions for the coordinates and expressing it in terms of this parametrization.

We now exposit in detail Algorithm 1 and its proof of correctness, in the case where f is a polynomial map; for simplicity we take $k = \mathbb{R}$. The argument is similar for smooth or holomorphic maps, except that one needs an additional theorem to arrive at the expression (7) below. If G is a compact Lie group, the needed theorem is proven in [14] for smooth maps, and in [10] for holomorphic maps over $\mathbb{C}$.

We begin with linear representations V and W of a group G over $\mathbb{R}$. We take W^* to be the contragredient

Algorithm 1 Malgrange's method for getting equivariant functions

Input: Bihomogeneous generators $f_1, \dots, f_m$ for $\mathbb{R}[V \times W^*]^G$.

1. Order the generators so that $f_1, \dots, f_r$ are of degree 0 and $f_{r+1}, \dots, f_s$ are of degree 1 in W^*. Discard $f_{s+1}, \dots, f_m$ (of higher degree in W^*).
2. Choose a basis $e_1, \dots, e_d$ for W, and let $e_1^\top, \dots, e_d^\top$ be the dual basis, so an arbitrary element $\ell \in W^*$ can be written
$$\ell = \sum_{i=1}^{d} \ell_i e_i^\top,$$
and $\ell(e_i) = \ell_i$.
3. For $j = r+1, \dots, s$, and for $v \in V, \ell \in W^*$, let $F_j(v)$ be the gradient of $f_j(v, \ell)$ with respect to $\ell \in W^*$, identified with an element of W along the canonical isomorphism $W^{**} \cong W$; explicitly,
$$F_j(v) := \sum_{i=1}^{d} \left(\frac{\partial}{\partial \ell_i} f_j(v, \ell) \right) e_i.$$
Then each F_j is a function $V \to W$.

Output: Equivariant polynomial functions $F_{r+1}, \dots, F_s$ from V to W such that any equivariant polynomial (or smooth, if G is compact) map $f : V \to W$ can be written as
$$f = \sum_{j=r+1}^{s} p_j(f_1, \dots, f_r) F_j,$$
where $p_j : \mathbb{R}^r \to \mathbb{R}$ are arbitrary polynomial (or smooth, if G is compact) functions.

Remark: Because the $f_j(v, \ell)$'s are being differentiated with respect to variables in which they are linear, Step 3 could alternatively have been stated: for each j, define F_j as the vector in W whose coefficients with respect to the e_i's are just the coefficients of the the ℓ_i's in $f_j(v, \ell)$.

representation to W, defined above. (If G is compact, we can work in a coordinate system in which the action of G on W is orthogonal, and then we may ignore the distinction between W and W^*, as discussed above in the case of $G = \mathrm{O}(d)$.) We suppose we have an explicit set $f_1, \dots, f_m$ of polynomials that generate the algebra of invariant polynomials on the vector space $V \times W^*$ (denoted as $\mathbb{R}[V \times W^*]^G$)—in other words, they have the property that *any* invariant polynomial can be written as a polynomial in these. We also assume they are *bihomogeneous*, i.e., independently homogeneous in V and in W^*. To reduce notational clutter we suppress the maps specifying the actions of G on V and W (which were called φ and ψ in the previous section), writing the image of $v \in V$ (respectively $w \in W$, $\ell \in W^*$) under the action of an element $g \in G$ as gv (respectively gw, $g\ell$). We suppose $f_1, \dots, f_r$ are degree 0

[1] *The action of G on $V \times W^*$ is defined by acting separately on each factor; the action on W^* is the contragredient representation defined above.*

in ℓ (so they are functions of v alone), $f_{r+1}, \dots, f_s$ are degree 1 in ℓ, and $f_{s+1}, \dots, f_m$ are degree > 1 in ℓ.

Now we consider an arbitrary G-equivariant polynomial function

$$f : V \to W.$$

We let $\ell \in W^*$ be arbitrary, and, as in the previous section, we construct the function

$$\begin{aligned} \tilde{f} &: V \times W^* \to \mathbb{R} \\ &(v, \ell) \mapsto \ell(f(v)). \end{aligned}$$

Equivariance of $f : V \to W$ implies that $\tilde{f}$ is invariant:

$$\tilde{f}(gv, g\ell) = \ell \circ g^{-1}(f(gv)) = \ell \circ g^{-1}(gf(v)) = \ell(f(v)) = \tilde{f}(v, \ell).$$

From the invariance of $\tilde{f}$, and the fact that $f_1, \dots, f_m$ generate the algebra of invariant polynomials on $V \times W^*$, we have an equality of the form

$$\tilde{f}(v, \ell) = P(f_1(v), \dots, f_m(v, \ell)), \tag{7}$$

where $P \in \mathbb{R}[X_1, \dots, X_m]$ is a polynomial. Note that $f_1, \dots, f_r$ do not depend on ℓ, while $f_{r+1}, \dots, f_m$ do.

We now fix $v \in V$ and take the gradient D_ℓ of both sides of (7) with respect to $\ell \in W^*$, viewed as an element of W.[2] Choosing dual bases $e_1, \dots, e_d$ for W and $e_1^\top, \dots, e_d^\top$ for W^*, and writing $\ell = \sum \ell_i e_i^\top$, we can express the operator D_ℓ acting on a smooth function $F : W^* \to \mathbb{R}$ explicitly by the formula

$$D_\ell F = \sum_{i=1}^{d} \left(\frac{\partial}{\partial \ell_i} F \right) e_i.$$

Applying D_ℓ to the left side of (7), we get

$$\begin{aligned} D_\ell \tilde{f}(v, \ell) &= \sum_{i=1}^{d} \left(\frac{\partial}{\partial \ell_i} \ell(f(v)) \right) e_i \\ &= \sum_{i=1}^{d} \left(\frac{\partial}{\partial \ell_i} \left(\sum_{j=1}^{n} \ell_j e_j^\top f(v) \right) \right) e_i \\ &= \sum_{i=1}^{d} \left(e_i^\top f(v) \right) e_i \\ &= f(v), \end{aligned}$$

so D_ℓ recovers f from $\tilde{f}$. (Indeed, this was the point.) Meanwhile, applying D_ℓ to the right side of (7), writing $\partial_j P$ for the partial derivative of P with respect to its jth argument, and using the chain rule, we get

$$D_\ell P(f_1, \dots, f_m) = \sum_{j=1}^{m} \partial_j P(f_1, \dots, f_m) D_\ell f_j.$$

Combining these, we conclude

$$f(v) = \sum_{j=1}^{m} \partial_j P(f_1(v), \dots, f_m(v, \ell)) D_\ell f_j(v, \ell). \tag{8}$$

[2]*In the background, we are using canonical isomorphisms to identify W^* with all its tangent spaces, and W^{**} with W.*

Now we observe that $D_\ell f_j(v) = 0$ if $j \le r$, because in those cases $f_j(v)$ is constant with respect to ℓ. But meanwhile, the left side of (8) does not depend on ℓ, and it follows the right side does not either; thus we can evaluate it at our favorite choice of ℓ; we take $\ell = 0$. Upon doing this, $D_\ell f_j(v, \ell)|_{\ell=0}$ also becomes 0 for $j > s$, because in these cases $f(v, \ell)$ is homogeneous of degree at least 2 in ℓ, so its partial derivatives with respect to the ℓ_i remain homogeneous degree at least 1 in ℓ, thus they vanish at $\ell = 0$. Meanwhile, $f_j(v, \ell)|_{\ell=0}$ itself vanishes for $j = r+1, \dots, m$, so that the $(r+1)$st to mth arguments of each $\partial_j P$ vanish. Abbreviating

$$\partial_j P(f_1(v), \dots, f_r(v), 0, \dots, 0)$$

as $p_j(f_1(v), \dots, f_r(v))$, we may thus rewrite (8) as

$$f(v) = \sum_{j=r+1}^{s} p_j(f_1(v), \dots, f_r(v)) D_\ell f_j(v, \ell).$$

Finally, we observe that, f_j being linear homogeneous in ℓ for $j = r+1, \dots, s$, $D_\ell f_j(v, \ell)$ is degree 0 in ℓ, i.e., it does not depend on ℓ. So we may call it $F_j(v)$ as in the algorithm, and we have finally expressed f as the sum $\sum_{r+1}^{s} p_j(f_1, \dots, f_r) F_j$, as promised.

6. Examples

In this section we apply Malgrange's method to parametrize equivariant functions in various examples. In all cases, for positive integers d, n, we take a group G of $d \times d$ matrices, equipped with its canonical action on $\mathbb{R}^d$, and we are looking for equivariant maps

$$(\mathbb{R}^d)^n \to \mathbb{R}^d$$

from an n-tuple of vectors to a single vector. The underlying invariant theory is provided by Weyl's *The Classical Groups* in each case.

The orthogonal group. We parametrize maps that are equivariant for $G = \mathrm{O}(d)$. By the Riesz representation theorem, we can identify $\mathbb{R}^d$ with $(\mathbb{R}^d)^*$ along the map $v \mapsto \langle v, \cdot \rangle$, where $\langle \cdot, \cdot \rangle$ is the standard dot product $\langle v, w \rangle = v^\top w$. Since $\mathrm{O}(d)$ preserves this product (by definition), this identification is equivariant with respect to the $\mathrm{O}(d)$-action, thus $(\mathbb{R}^d)^*$ is isomorphic with $\mathbb{R}^d$ as a representation of $\mathrm{O}(d)$. We may therefore ignore the difference between $W = \mathbb{R}^d$ and W^* in applying the algorithm.

Thus, we consider the ring of $\mathrm{O}(d)$-invariant polynomials on tuples

$$(v_1, \dots, v_n, \ell) \in (\mathbb{R}^d)^n \times \mathbb{R}^d = V \times W.$$

We begin with bihomogeneous generators for this ring. By a classical theorem of Weyl known as the *first fundamental theorem for* $\mathrm{O}(d)$, they are the dot products $v_i^\top v_j$ for $1 \le i \le j \le n$; $v_i^\top \ell$ for $1 \le i \le n$; and $\ell^\top \ell$. These are ordered by their degree in W as in Step 1 of the algorithm; we discard $\ell^\top \ell$ as it is degree > 1 in W.

We can work in the standard basis $e_1, \dots, e_d$ for $W = \mathbb{R}^d$, and we have identified it with its dual, so Step 2 is done as well.

Applying Step 3, we take the generators of degree 1 in W, which are

$$f_1 = v_1^\top \ell, \dots, f_n = v_n^\top \ell.$$

Taking the gradients, we get

$$\begin{aligned} F_j(v_1, \dots, v_n) &= \sum_{i=1}^d \left(\frac{\partial}{\partial \ell_i} v_j^\top \ell \right) e_i \\ &= \sum_{i=1}^d (v_j)_i e_i \\ &= v_j, \end{aligned}$$

where $(v_j)_i$ denotes the ith coordinate of v_j. Thus the $F_j(v_1, \dots, v_n)$ yielded by the algorithm is nothing but projection to the jth input vector. Meanwhile, the $f_1(v), \dots, f_r(v)$ of the algorithm are the algebra generators $v_i^\top v_j$ of degree zero in ℓ; thus the output of the algorithm is precisely the representation described in (5) and the paragraph following.

The Lorentz and symplectic groups. If we replace $\mathrm{O}(d)$ with the Lorentz group $\mathrm{O}(1, d-1)$, or (in case d is even) the symplectic group $\mathrm{Sp}(d)$, the entire discussion above can be copied verbatim, except with the standard dot product being replaced everywhere by the Minkowski product $v^\top \operatorname{diag}(-1, 1, \dots, 1) v$ in the former case, or the standard skew-symmetric bilinear form $v^\top J w$ (where J is block diagonal with 2×2 $\pi/2$-rotation matrices as blocks) in the latter. We also need to use these respective products in place of the standard dot product to identify $\mathbb{R}^d$ equivariantly with its dual representation. The key point is that the invariant theory works the same way (see [12, Sec. 9.3] for a concise modern treatment, noting that $O(d)$ and $O(1, d-1)$ have the same complexification).

The special orthogonal group. Now we consider $G = \mathrm{SO}(d)$. We can once again identify $\mathbb{R}^d$ with its dual. However, this time, in Step 1, the list of bihomogeneous generators is longer: in addition to the dot products $v_i^\top v_j$ and $v_i^\top \ell$ (and $\ell^\top \ell$, which will be discarded), we have $d \times d$ determinants $\det(v_{i_1}, \dots, v_{i_d})$ for $1 \le i_1 \le \cdots \le i_d \le n$, and $\det(v_{i_1}, \dots, v_{i_{d-1}}, \ell)$ for $1 \le i_1 \le \cdots \le i_{d-1} \le n$. The former are of degree 0 in ℓ while the latter are of degree 1. Thus, the latter contribute to our list of F_j's in Step 3, while the former figure in the arguments of the p_j's. Carrying out Step 3 in this case, we find that

$$\sum_{i=1}^d \left(\frac{\partial}{\partial \ell_i} \det(v_{i_1}, \dots, v_{i_{d-1}}, \ell) \right) e_i$$

is exactly the generalized cross product of the $d-1$ vectors $v_{i_1}, \dots, v_{i_{d-1}}$. Thus we must add to the F_j's a generalized cross-product for each $\binom{n}{d-1}$-subset of our input vectors; in the end the parametrization of equivariant maps looks like

$$\begin{aligned} f = &\sum_{i=1}^n p_i \left((v_j^\top v_k)_{\{j,k\} \in \binom{[n]}{2}}, \det(v|_S)_{S \in \binom{[n]}{d}} \right) v_i \\ &+ \sum_{S' \in \binom{[n]}{d-1}} p_{S'} \left((v_j^\top v_k)_{\{j,k\} \in \binom{[n]}{2}}, \det(v|_S)_{S \in \binom{[n]}{d}} \right) v_{S'}, \end{aligned}$$

where $[n] := \{1, \dots, n\}$, $\binom{[n]}{k}$ represents the set of k-subsets of $[n]$, $\det(v|_S)$ is shorthand for $\det(v_{i_1}, \dots, v_{i_d})$ where $S = \{i_1, \dots, i_d\}$, and $v_{S'}$ is shorthand for the generalized cross product of the $d-1$ vectors $v_{i_1}, \dots, v_{i_{d-1}}$ where $S' = \{i_1, \dots, i_{d-1}\}$.

The special linear group. We include an example where we cannot identify $\mathbb{R}^d$ with its dual representation. Namely, we take $G = \mathrm{SL}(d, \mathbb{R})$. As G does not preserve any bilinear form on $\mathbb{R}^d$, we must regard $(\mathbb{R}^d)^*$ as a distinct representation. Thus in Step 1 we must consider the polynomial invariants on

$$(v_1, \dots, v_n, \ell) \in (\mathbb{R}^d)^n \times (\mathbb{R}^d)^*.$$

A generating set of homogeneous invariants is given by the canonical pairings $\ell(v_i)$, $i = 1, \dots, n$, and the $d \times d$ determinants $\det(v|_S)$, $S \in \binom{[n]}{d}$, where we have used the same shorthand as above for a d-subset S of $[n]$ and the $d \times d$ determinant $\det(v|_S)$ whose columns are the v_i's indexed by S. The former are degree 1 in ℓ while the latter are degree 0. So, writing $\ell = \sum_{i=1}^d \ell_i e_i^\top$ as in Step 2, we have

$$f_{r+j}(v, \ell) = \ell(v_j) = \sum_{i=1}^d \ell_i \cdot (v_j)_i,$$

and again in Step 3 we get

$$F_j(v) = v_j,$$

with the computation identical to the one above for the orthogonal group. Thus the algorithm outputs that an arbitrary $G = \mathrm{SL}(d, \mathbb{R})$-equivariant polynomial map has the form

$$f = \sum_{j=1}^n p_j \left(\det(v|_S)_{S \in \binom{[n]}{d}} \right) v_j.$$

The symmetric group. We give an example where the algebra of invariants is generated by something other than determinants and bilinear forms, so the F_j's output by the algorithm are not just the v_j's themselves and generalized cross products. Take $G = S_d$, the symmetric group on d letters, acting on $\mathbb{R}^d$ by permutations of the coordinates. As S_d realized in this way is a subgroup of $\mathrm{O}(d)$, we can once again identify $\mathbb{R}^d$ with its dual.

By the fundamental theorem on symmetric polynomials, the algebra of G-invariant polynomials on a single vector $v \in \mathbb{R}^d$ is given by the elementary symmetric polynomials in the coordinates $\sigma_1(v) = \sum(v)_\alpha$, $\sigma_2(v) = \sum_{\alpha<\beta}(v)_\alpha(v)_\beta$, etc., where $\alpha, \beta, \dots \in [d]$ index the coordinates of the vector v. Weyl showed that for any n, the algebra of invariant polynomials on an n-tuple of vectors $(v_1, \dots, v_n) \in (\mathbb{R}^d)^n$, is generated by the *polarized* elementary symmetric polynomials

$$\sigma_1(v_i) = \sum_\alpha (v_i)_\alpha,$$
$$\sigma_2(v_i, v_j) = \sum_{\alpha\neq\beta} (v_i)_\alpha (v_j)_\beta,$$
$$\sigma_3(v_i, v_j, v_k) = \sum_{\alpha\neq\beta\neq\gamma} (v_i)_\alpha (v_j)_\beta (v_k)_\gamma,$$

etc., where $v_i, v_j, v_k, \dots$ can be any of the vectors $v_1, \dots, v_n$, distinct or not. (Up to a scalar multiple, one recovers the original, unpolarized elementary symmetric polynomials of a single vector by setting $i = j = k = \dots$.) Running the algorithm to parametrize equivariant functions $(\mathbb{R}^d)^n \to \mathbb{R}^d$, we write down the algebra of invariants on $(v_1, \dots, v_n, \ell) \in (\mathbb{R}^d)^n \times \mathbb{R}^d$, and see that the algebra generators of degree 1 in ℓ have the form $\sigma_s(v_{j_1}, \dots, v_{j_{s-1}}, \ell)$, where, again, $j_1, \dots, j_{s-1} \in [n]$ can be distinct or not. The gradients of these become the F_j's of Step 3. For the sake of explicitness, we fix $d = 3$, $n = 2$, and write out the results as column vectors. We get

$$D_\ell\sigma_1(\ell) = \begin{pmatrix}1\\1\\1\end{pmatrix}; D_\ell\sigma_2(v_i, \ell) = \begin{pmatrix}(v_i)_2 + (v_i)_3\\(v_i)_1 + (v_i)_3\\(v_i)_1 + (v_i)_2\end{pmatrix}, i = 1, 2;$$

$$D_\ell\sigma_3(v_i, v_j, \ell) = \begin{pmatrix}(v_i)_2(v_j)_3 + (v_i)_3(v_j)_2\\(v_i)_1(v_j)_3 + (v_i)_3(v_j)_1\\(v_i)_1(v_j)_2 + (v_i)_2(v_j)_1\end{pmatrix}, (i, j) = (1, 1), (1, 2), (2, 2).$$

Thus any S_3-equivariant polynomial map $(\mathbb{R}^3)^2 \to \mathbb{R}^3$ is a linear combination of these six maps, with coefficients that are polynomials in $\sigma_1(v_i)$, $\sigma_2(v_i, v_j)$, and $\sigma_3(v_i, v_j, v_k)$ with $i, j, k \in \{1, 2\}$.

7. Discussion

This article gives a gentle introduction to equivariant machine learning, and it explains how to parameterize equivariant maps $V \to W$ in two different ways. One requires knowledge of the irreducible representations and Clebsch-Gordan coefficients, and the other requires the knowledge of the generators of the algebra of invariant polynomials on $V \times W^*$. The main focus is on the latter, which is useful to design equivariant machine learning with respect to groups and actions where the invariants are known and are computationally tractable. This is not a panacea: sometimes the algebra of invariant polynomials is too large, or outright not known. Both these issues come up, for example, in the action of permutations on $n \times n$ symmetric matrices by conjugation, the relevant one for graph neural networks. The invariant ring has not been fully described as of this writing, except for small n, where the number of generators increases very rapidly with n; see [17].

From a practitioner's point of view, it is not yet clear which of these approaches will behave better in practice. We conjecture that Malgrange's construction applied to nonpolynomial invariants such as [2, 5, 11], as described at the end of Section 3, is a promising direction for some applications, especially because some of them exhibit desirable stability properties.

ACKNOWLEDGMENTS. The authors thank Gerald Schwarz (Brandeis University) for introducing us to Malgrange's method. The authors also thank Jaume de Dios Pont (UCLA), David W. Hogg (NYU), Teresa Huang (JHU), and Peter Olver (UMN) for useful discussions. SV and BBS are partially supported by ONR N00014-22-1-2126. SV was also partially supported by the NSF-Simons Research Collaboration on the Mathematical and Scientific Foundations of Deep Learning (MoDL) (NSF DMS 2031985), NSF CISE 2212457, and an AI2AI Amazon research award.

References

[1] B. Blum-Smith and S. Villar, *Machine learning and invariant theory*, arXiv preprint arXiv:2209.14991, 2023.

[2] J. Cahill, J. W. Iverson, D. G. Mixon, and D. Packer, *Group-invariant max filtering*, arXiv preprint arXiv:2205.14039, 2022.

[3] T. Cohen and M. Welling, *Group equivariant convolutional networks*, Proceedings of the 33rd International Conference on Machine Learning, PMLR, 2990–2999, 2016.

[4] Harm Derksen and Gregor Kemper, *Computational invariant theory*, Second enlarged edition, Encyclopaedia of Mathematical Sciences, vol. 130, Springer, Heidelberg, 2015. With two appendices by Vladimir L. Popov, and an addendum by Norbert A'Campo and Popov; Invariant Theory and Algebraic Transformation Groups, VIII, DOI 10.1007/978-3-662-48422-7. MR3445218

[5] N. Dym and S. J. Gortler, *Low dimensional invariant embeddings for universal geometric learning*, arXiv preprint arXiv:2205.02956, 2022.

[6] N. Dym and H. Maron, *On the universality of rotation equivariant point cloud networks*, International Conference on Learning Representations, 2021.

[7] Karin Gatermann, *Computer algebra methods for equivariant dynamical systems*, Lecture Notes in Mathematics, vol. 1728, Springer-Verlag, Berlin, 2000, DOI 10.1007/BFb0104059. MR1755001

[8] M. Geiger and T. Smidt, *e3nn: Euclidean neural networks*, arXiv preprint arXiv:2207.09453, 2022.

[9] R. Kondor, Z. Lin, and S. Trivedi, *Clebsch–Gordan nets: a fully fourier space spherical convolutional neural network*, Advances in Neural Information Processing Systems **31** (2018), 10117–10126.

[10] Domingo Luna, *Fonctions différentiables invariantes sous l'opération d'un groupe réductif* (French, with English summary), Ann. Inst. Fourier (Grenoble) **26** (1976), no. 1, ix, 33–49. MR423398

[11] Peter J. Olver, *Invariants of finite and discrete group actions via moving frames*, Bull. Iranian Math. Soc. **49** (2023), no. 2, Paper No. 11, 12, DOI 10.1007/s41980-023-00744-0. MR4549776

[12] V. L. Popov and E. B. Vinberg. Invariant theory. In *Algebraic geometry IV*, pages 123–278. Springer, 1994.

[13] F. Ronga, *Stabilité locale des applications équivariantes*, Differential topology and geometry (Proc. Colloq., Dijon, 1974), Lecture Notes in Math., Vol. 484, Springer, Berlin, 1975, pp. 23–35. MR0445526

[14] Gerald W. Schwarz, *Smooth functions invariant under the action of a compact Lie group*, Topology **14** (1975), 63–68, DOI 10.1016/0040-9383(75)90036-1. MR370643

[15] Gerald W. Schwarz, *Lifting smooth homotopies of orbit spaces*, Inst. Hautes Études Sci. Publ. Math. **51** (1980), 37–135. MR573821

[16] J. Shawe-Taylor, *Building symmetries into feedforward networks*, 1989 First IEE International Conference on Artificial Neural Networks (Conf. Publ. No. 313), 158–162, IET, 1989.

[17] N. M. Thiéry, *Algebraic invariants of graphs; a study based on computer exploration*, ACM SIGSAM Bulletin **34** (2000), no. 3, 9–20.

[18] S. Villar, D. W. Hogg, K. Storey-Fisher, W. Yao, and B. Blum-Smith, *Scalars are universal: Equivariant machine learning, structured like classical physics*, Advances in Neural Information Processing Systems **34** (2021), 28848–28863.

[19] S. Villar, D. W. Hogg, W. Yao, G. A. Kevrekidis, and B. Schölkopf, *The passive symmetries of machine learning*, arXiv preprint arXiv:2301.13724, 2023.

[20] Patrick A. Worfolk, *Zeros of equivariant vector fields: algorithms for an invariant approach*, J. Symbolic Comput. **17** (1994), no. 6, 487–511, DOI 10.1006/jsco.1994.1031. MR1300350

Ben Blum-Smith

Soledad Villar

Credits

Opening image is courtesy of filo via Getty.
Photo of Ben Blum-Smith is courtesy of Ryan Lash/TED.
Photo of Soledad Villar is courtesy of Erwin List/Soledad Villar.