Machine Learning: Mathematical Theory and Scientific Applications

Weinan E

This is essentially the transcript of the Peter Henrici Prize Lecture, given on July 15 at ICIAM 2019 in Valencia. The talk was designed to give a broad overview of

- some of the current work on integrating machine learning with scientific modeling (e.g. physics-based modeling) to address some of the most challenging problems in a variety of disciplines, and
- some of the current work on building a mathematical theory of machine learning.

There were two basic messages that I wanted to convey. First, in theoretical and computational science and engineering, a fundamental obstacle that we have encountered is our limited ability to deal with problems in high dimension. Machine learning has now provided new tools for handling this problem. The integration of machine learning and scientific modeling will give us unprecedented technical power and will likely change the way we do science and engineering.

Secondly, even though until now machine learning has not been the most popular area in applied mathematics, the spirit of machine learning is very much aligned with numerical analysis, except for the fact that machine learning has to be concerned with problems in very high dimension. To build the theoretical foundation of machine learning, we need to develop high-dimensional numerical analysis.

PDEs and Fundamental Laws of Physics

One of the most important roles of mathematics is that it provides the language with which the fundamental laws of physics are formulated. These laws are often expressed in terms of partial differential equations (PDEs). Two of the most important such PDEs are the Schrödinger equation for quantum mechanics and the Navier–Stokes equations for fluid mechanics. Other examples include the Boltzmann equation for kinetic theory, the linear and nonlinear elasticity equations, and Maxwell's equation for electromagnetism.

Weinan E is a professor in the Department of Mathematics and the Program in Applied and Computational Mathematics at Princeton University. His email address is weinan@math.princeton.edu.

This work was supported in part by the Office of Naval Research and iFlytek through a gift to Princeton University.

Communicated by Notices *Associate Editor Reza Malek-Madani.*

DOI: https://doi.org/10.1090/noti1994

Back in 1929 when quantum mechanics was just established, Paul Dirac made the following observation [7]:

"The underlying physical laws necessary for the mathematical theory of a large part of physics and the whole of chemistry are thus completely known, and the difficulty is only that the exact application of these laws leads to equations much too complicated to be soluble."

Basically what Dirac was saying was that for most situations encountered in practice, the difficulty no longer lies in the physical problem of finding the fundamental laws, but in the mathematical problem of solving the PDEs that express these laws.

Previous Successes and Failures

In the years immediately following the establishment of these fundamental laws, efforts were focused on using analytical methods to find either approximate models or approximate solutions. For example, density functional theory (DFT), a much simplified model for quantum mechanics, was proposed in the same year by Fermi and Thomas [13]. Many years later this theory was developed into a workhorse for computational materials science and chemistry [13], and one of the main contributors, Walter Kohn, was awarded the Nobel Prize in chemistry in 1998. Even though they are heavily used in practice, many such approximate models are quite ad hoc and often involve uncontrolled approximations. In the applied mathematics community, asymptotic methods were developed to find approximate solutions, with mixed success [15].

Solving differential equations numerically. The first general approach for solving these PDEs came during the 1950s after modern computers were born. During this period and the time afterwards, numerical algorithms, particularly finite difference and finite element methods, were developed, analyzed, and applied to a wide variety of problems. This has been a very successful activity, continuing even to this day. These numerical methods are so powerful that they have become standard tools in engineering and some disciplines in science. Many problems have been essentially solved, including problems in gas dynamics, structural analysis, radar, sonar, etc.

Yet many problems remain difficult. Examples of these problems include: classical or quantum many-body problems, first principle-based drug and materials design, protein folding, turbulence, plasticity, and non-Newtonian fluid. A common feature of these problems is that the solutions depend on many variables in an essential way. For example, even though turbulence as described by the Navier–Stokes equations is a three-dimensional problem, its solutions contain many active scales, which means that many degrees of freedom are involved. What makes these problems difficult is the *curse of dimensionality*: As the dimension (i.e., the number of variables or degrees of freedom) grows, the complexity (or computational cost) grows exponentially. This has been an essential obstacle across a wide spectrum of applications.

Multiscale modeling. The second important advance was the development of multiscale, multiphysics algorithms. Here what we are interested in is modeling some macro-scale system, but we lack reliable physical models at that scale. Instead, we have a reliable micro-scale model at a finer scale that involves many more degrees of freedom. The idea is to develop algorithms that are capable of modeling macro-scale systems but using only micro-scale models. Notable examples of such methodologies include the heterogeneous multiscale method (HMM), the equation-free approach, and the quasi-continuum method [8]. HMM starts out with a prescribed form of a macro-scale model involving unknown quantities and uses the micro-scale model to estimate the values of these quantities "on the fly" as the computation proceeds. For problems for which there is a scale separation between the macro- and micro-scale processes, this kind of approach has been quite successful. However, success has been quite limited for problems that lack scale separation.

It was already realized in the early days of HMM that the difficulty was in the estimation of unknown quantities in the macro-scale model using data from micro-scale simulations, particularly for problems without scale separation. Suppose we want to model the large scale dynamics of turbulent flows and we use the large eddy simulation model as our macro-scale model. The unknown quantities here are the Leonard stresses, the contribution to the averaged stress from the unresolved degrees of freedom, which have to be estimated from the original Navier–Stokes equations. The difficulty comes from the fact that the Leonard stress depends on many degrees of freedom; therefore estimating this stress from the Navier–Stokes equations is a very difficult task.

Integrating Machine Learning with First-Principle-Based Modeling

The main source of difficulty in all these problems comes from the curse of dimensionality, namely, our limited ability for dealing with functions of many variables. In recent years, we have seen the success of machine learning, particularly deep learning, in attacking problems of this type in computer vision and other artificial intelligence (AI) tasks. It is natural to ask whether machine learning can be of help in overcoming similar obstacles in the kinds of applications described above or, more generally, applications outside the realm of traditional AI. In the following we will describe some examples of initial successes in this direction.

Before proceeding further, we should point out some differences between machine learning tasks for the kinds of applications we are concerned with here and for traditional AI tasks. The first difference is that here we are interested in using machine learning to help come up with reliable and practical physical models. These models should not violate the constraints from physics such as symmetry and invariants. The second difference is that oftentimes the data that we use to train the machine learning models are generated by some (micro-scale) physical model. In principle we can generate an infinite amount of data, but in practice the process of generating data is typically quite expensive. Therefore there is a need for algorithms that can deliver an "optimal set of data": the data set should cover all practical situations we are interested in but should otherwise be as small as possible. This falls into the framework of *active learning*. This can also be thought of as being an adaptive algorithm in which the sampling of the data is done adaptively. To develop machine learning-based physical models, we have to address these two general issues.

Molecular dynamics. The objective of molecular dynamics is to model a molecular or a material system by following the trajectory of all the atomic nuclei in the system. To this end, it is a reasonable approximation to use classical Newtonian dynamics, with an interatomic potential defined by the position of all of the participating nuclei and the associated electronic structure. Modeling this interatomic potential function, a function of many variables—also known as the potential energy surface, or PES—is a major difficulty in molecular modeling. There have been two different approaches. One is to evaluate "on the fly" this function using quantum mechanical models, usually the DFT. This is known broadly as *ab initio* molecular dynamics (AIMD) (see [8]). The second is to construct this function empirically using a guessed functional form and some limited amount of experimental and numerical data. The first approach is quite accurate but very expensive, limiting the size of the systems that one can model to a few hundred atoms. The second approach is much more efficient but unreliable in terms of accuracy.

With the help of machine learning, one can envision a new paradigm for molecular dynamics. In this new paradigm, quantum mechanical models serve as a generator for the data, from which machine learning is used to parametrize the PES. This new approach can potentially provide a way for performing molecular dynamics with accuracy comparable to AIMD but with complexity comparable to that of the empirical approach.

Concerning the two general issues raised earlier, the important symmetries for this problem are the translational, rotational, and permutational symmetries. The last refers to the fact that if we relabel the atoms of the same species, the system does not change; therefore the PES should also remain unchanged.

To ensure invariance under these symmetries as well as scalability, the following designing principles have been proposed for the neural network structure [5, 17]:

1. The full network is the superposition of subnetworks, each of which corresponds to one nucleus in the system. This ensures scalability. However, this kind of structure is not suited for representing long-range interactions.
2. Each subnetwork consists of an encoding network, followed by a fitting network. The encoding network ensures that the data going into the fitting network satisfy the symmetry constraints.

The active learning algorithm developed in [18] consists of three major components:

- exploration:
  - macro: Sample the space of thermodynamic variables, e.g., temperature and pressure.
  - micro: For each fixed set of thermodynamic variables, sample the corresponding canonical ensemble.

  Both can be done by standard approaches.
- labeling:
  - Decide whether a particular atomic configuration should be labeled, using an error estimator. One simple error estimator is given as follows. Train an ensemble of neural network models (for example, all with the same network structure but trained with different initialization), and use the variance of their predictions as the error estimator. Large values of the error estimator imply that the current network model is not accurate enough for the configuration considered. Therefore this configuration needs to be labeled and put in the data set for retraining.
  - For each configuration that needs to be labeled, evaluate its potential energy and the forces on the nuclei using DFT, and then put the results in the training data set.
- training: Update the network parameters by fitting to the energies and forces in the data set.

Once the system of interest is specified, one can start out with essentially no data, and the algorithm will keep improving the model until satisfactory accuracy is achieved. Examples can be found in [18]. It was found that only about .01% of configurations explored need to be labeled.

By now this set of ideas has been applied to a wide variety of systems, including small and big molecules, water, semiconductors, surface materials, high-entropy alloys, etc. In each case, one can obtain machine learning models with accuracy comparable to that of the DFT. A general purpose software called DeePMD-kit has also been developed and has already been used by a number of groups worldwide [16].

Modeling gas dynamics. The dynamics of gases is modeled by the Boltzmann equation that describes the evolution of the one-particle phase space distribution function. For dense gases, the Boltzmann equation can be approximated accurately by Euler's equation that describes the evolution of density, momentum, and energy profiles in space. A crucial quantity that controls this approximation is the Knudsen number, which is the ratio between the mean-free-path of the gas and a typical length scale of the system. Euler's equation is an accurate approximation when the Knudsen number is small. In this case the one-particle distribution function stays close to the so-called local Maxwellian, and Euler's equation is obtained by projecting the Boltzmann equation onto its 0th, 1st, and the trace of the 2nd order moments [2].

There has been a lot of effort devoted to extending the validity of the Euler or Euler-like equations for larger values of the Knudsen number by including more moments in the projection scheme. This effort has suffered two major difficulties [6]:

1. The moment equations obtained are not well-posed. For example, it is well known that Grad's 13 moment equations fail to be hyperbolic in certain regions of the state space.
2. The closure problem: To obtain closed equations, one needs to approximate the higher order moment terms that appear in the projected system. For larger values of the Knudsen number, one can no longer use the local Maxwellian as an ansatz to close the system.

Machine learning offers some hope for constructing (generalized) moment models that are uniformly accurate over a wide range of Knudsen numbers, as has been demonstrated in [12]. This is done in two steps:

1. Learn a set of generalized moments that best represent the distribution function. One way for doing this is through an auto-encoder that minimizes the reconstruction error for the distribution function.
2. Learn the dynamics for this set of generalized moments. The terms that appear in the dynamic equation can all be expressed in terms of the distribution function. Approximations to these terms can be found through supervised learning.

The main issues are again how to preserve physical symmetries and how to obtain the data set through the microscopic model, here the Boltzmann equation. Compared with the previous example, here we have a new dynamic symmetry, the Galilean invariance. In fact, in terms of machine learning, the previous example is a more conventional example of learning a function; the current example is about learning a new dynamical system. Preliminary tests have shown that this approach holds a lot of promise [12]. This is also an example of multiscale modeling for problems without scale separation.

High-dimensional partial differential equations. Solving PDEs in high dimensions is a classical example that suffers from the curse of dimensionality. One exception is found in linear parabolic PDEs. In this case, we can express the solution as an expectation of a functional of the Brownian path using the Feynman–Kac formula [14] and apply the Monte Carlo method to evaluate the solution. For nonlinear parabolic PDEs, the analog of the Feynman–Kac formula is expressed in terms of the backward stochastic differential equations (BSDEs) [14]. This allows us to formulate an algorithm for solving nonlinear parabolic equations in which the gradients of the solution at a discrete set of time slices are approximated using neural networks. The BSDE formula (actually its discrete version) is then used to evaluate the solution at a prescribed space-time location, and the discrepancy between the approximate solution and the given terminal (or initial) condition is used as the loss function to train the network parameters [9]. This algorithm has turned out to be highly successful and can be used to solve a wide class of nonlinear parabolic PDEs and BSDEs in rather high dimensions. One byproduct is that the BSDE, an elegant mathematical tool for finance, economics, stochastic control, and other applications, has now become a powerful practical tool.

One application of these ideas has been found in nonlinear Black–Scholes equations for option pricing, say with default risks.

Mathematical principles of natural languages. The success of machine learning has triggered some friction between linguistics and natural language processing (NLP). On one hand, the fact that NLP has achieved much more impressive success in machine translation and other practical tasks has raised questions about the practical value of linguistics. On the other hand, doubts have also been raised about the efficiency of the machine learning approach since it requires a huge amount of training data, far more than what is required by a human, so it seems. Moreover, the black-box nature of NLP algorithms is also a concern. It is therefore desirable to develop quantitative structural models of natural languages that can help to bridge the gap between linguistics and NLP.

Aside from this, a mathematical model of natural languages has its own value. For example, an important question for both NLP and linguistics is the definition of semantics. To address such questions, one eventually has to turn to mathematical models.

Languages exhibit a number of scales such as words, sentences, and paragraphs. Different structural models are required at different scales. At smaller scales such as words and sentences, different languages exhibit a wide variety of diversity and irregularity. However, at longer scales, different languages exhibit remarkable universality. This universality is the basis for the fact that languages can be translated between each other [11].

A mathematically attractive definition of semantics is that it is the invariant after translation. If we view translation as operators between different languages, the fact that semantics is preserved after translation means that the generators for different languages are all similar to one another:

$$\mathbf{P}_{\mathrm{A}}\mathbf{T}_{\mathrm{A}\to\mathrm{B}} = \mathbf{T}_{\mathrm{A}\to\mathrm{B}}\mathbf{P}_{\mathrm{B}}.$$

Here $\mathbf{P}_{\mathrm{A}}$ and $\mathbf{P}_{\mathrm{B}}$ are symbolically the generators for languages A and B, respectively, and $\mathbf{T}_{\mathrm{A}\to\mathrm{B}}$ is the translation operator. Consequently the spectrum of $\mathbf{P}_{\mathrm{A}}$ and $\mathbf{P}_{\mathrm{B}}$ must be the same.

The challenge is to convert these intuitively attractive statements into mathematical models. An initial attempt has been made in [11]. Besides confirming the invariance properties described above, it also provided a quantitative structural model that seems to be able to capture the dynamics at some intersentence scale. One byproduct is that one can use this structural model to develop algorithms for machine translation with state-of-the-art performance but using much smaller training samples (see [11]).

Mathematical Theory of Machine Learning

We will focus on the problem of supervised learning for which a reasonably clear mathematical picture is emerging. But we believe the philosophy discussed here should be applicable to unsupervised learning and reinforcement learning.

Simply stated, the problem of supervised learning is to approximate a given target function using a finite sample of function values. Denote the target function by $f^* : \mathbb{R}^d \to \mathbb{R}^1$ and let $\{\mathbf{x}_j\}_{j=1}^n$ be a data set, independently sampled from a distribution μ on $\mathbb{R}^d$. Let $y_j = f^*(\mathbf{x}_j)$, $j = 1, \dots, n$. Since adding measurement noise typically does not change the argument in any essential way, we will neglect it for clarity of presentation. Our task is to approximate f^* using $S = \{(\mathbf{x}_j, y_j)\}_{j=1}^n$. This is done through the following steps:

1. Construct some "hypothesis space" (which is a set of functions) $\mathcal{H}_m = \{f(\cdot, \theta)\}$. Here m is roughly the dimension of $\mathcal{H}_m$, and θ denotes the parameters corresponding to a specific function in $\mathcal{H}_m$.
2. Minimize the "empirical risk" over the hypothesis space:

$$\begin{aligned}\hat{\mathcal{R}}_n(\theta) &= \frac{1}{n}\sum_j (f(\mathbf{x}_j, \theta) - y_j)^2 \\ &= \frac{1}{n}\sum_j (f(\mathbf{x}_j, \theta) - f^*(\mathbf{x}_j))^2.\end{aligned}$$

Typical examples of the hypothesis space include:

- linear regression: $f(\mathbf{x}, \theta) = \beta \cdot \mathbf{x} + \beta_0, \theta = (\beta, \beta_0)$.
- generalized linear models: $f(\mathbf{x}, \theta) = \sum_{k=1}^m c_k \phi_k(\mathbf{x}), \theta = (c_1, c_2, \dots, c_m)$, where $\{\phi_k\}$ is a set of linearly independent functions.
- two-layer neural networks: $f(\mathbf{x}, \theta) = \sum_k a_k \sigma(\boldsymbol{b}_k \cdot \mathbf{x} + c_k), \theta = \{a_k, \boldsymbol{b}_k, c_k\}$, where σ is some nonlinear scalar function, e.g., $\sigma(z) = \max(z, 0)$.
- deep neural networks (DNN): compositions of functions of the form above.

Approximating functions is a topic that has been very well studied in classical numerical analysis and approximation theory. The standard line of attack there is as follows:

- Define a "well-posed" mathematical model. This includes specifying the hypothesis space and the loss function. For example, in the case of one-dimensional cubic splines, the hypothesis space consists of C^1 piecewise cubic polynomials, and the loss function is defined by

$$I_n(f) = \frac{1}{n}\sum_{j=1}^n (f(x_j) - y_j)^2 + \lambda \int |f''(x)|^2 dx.$$

- Identify the right function spaces, e.g., Sobolev/Besov spaces. These comprise the space of functions for which the direct and inverse approximation theorems (also known as the Bernstein and Jackson type theorems) are valid; i.e., a function is in a certain function space if and only if it can be approximated by the given approximation scheme with a specified order of accuracy.
- Derive optimal error estimates. There are two kinds of error estimates. In a priori estimates, the error bounds depend on the norms of the target function. In a posteriori estimates, the error bounds depend on the norms of the numerical approximation. For example, for piecewise linear finite elements, typical a priori and a posteriori estimates take the form ($\alpha = 1/d, s = 2$):

$$\begin{aligned}\|f_m - f^*\|_{H^1} &\le Cm^{-\alpha}\|f^*\|_{H^s}, \\ \|f_m - f^*\|_{H^1} &\le Cm^{-\alpha}\|f_m\|_h.\end{aligned}$$

Here $\|\cdot\|_{H^s}$ is the sth order Sobolev norm, and $\|\cdot\|_h$ is typically a mesh-dependent norm [1].

There are two major differences between this classical setting and that of machine learning. The first is that in machine learning, we have to deal with very high dimension. We can already see that the estimates above suffer from the curse of dimensionality. We are interested in machine learning models that do not have this problem. The second is that in machine learning, we have only a finite amount of data. Therefore we can work only with the empirical risk. But what we are really interested in is the population risk:

$$\mathcal{R}(\theta) = \mathbb{E}(f(\mathbf{x},\theta) - f^*(\mathbf{x}))^2 = \int_{\mathbb{R}^d} (f(\mathbf{x},\theta) - f^*(\mathbf{x}))^2 d\mu.$$

The difference between the two, sometimes referred to as the generalization gap, is another key issue that we have to deal with. It turns out that the issue of the curse of dimensionality also arises here.

To appreciate these issues, let us examine a simple example of generalized linear model. In this case the hypothesis space is

$$f(\mathbf{x},\theta) = \sum_{k=1}^{m} a_k \phi_k(\mathbf{x}), \qquad \theta = (a_1, a_2, \cdots, a_m),$$

where $\{\phi_k\}$ is a set of linearly independent functions. Let us look at the case when $m > n$. In this case, to fit to data, i.e., to reduce the empirical risk to 0, its global minimum value, one only has to choose parameter θ that satisfies

$$G\theta^T = \boldsymbol{y},$$

where $G = (\phi_k(\mathbf{x}_j)), \boldsymbol{y} = (y_1, y_2, \dots, y_n)^T$. This is a linear system with more unknowns than equations. Unless G is degenerate, it has an infinite number of solutions. Yet one can prove that if θ is chosen as the solution of the above system with minimum Euclidean norm [4], then

$$\sup_{\|f\|_{\mathcal{B}_1} \le 1} \inf_{h \in \mathcal{H}_m} \|f - h\|_{L^2(D_0)} \ge \frac{C}{d m^{1/d}},$$

showing the curse of dimensionality. Here $\mathcal{B}_1$ is the Barron space defined in [10]. In this case, even though the empirical risk is 0, the population risk can still be quite large.

For high-dimensional problems, an important benchmark is the integration problem. Suppose we want to evaluate approximately the integral

$$I(g) = \int g(\mathbf{x}) d\mu.$$

It is well known that quadrature rules such as Simpson's rule suffer from the curse of dimensionality. However, the Monte Carlo method does not have this problem. Let $\{\mathbf{x}_j, j = 1, \dots, n\}$ be a set of independent random variables sampled from μ and let

$$I_n(g) = \frac{1}{n} \sum_{j=1}^{n} g(\mathbf{x}_j).$$

Then we have the identity

$$\mathbb{E}(I(g) - I_n(g))^2 = \frac{1}{n}\mathrm{var}(g), \tag{1}$$

where $\mathrm{var}(g) = \int_X g^2(\mathbf{x}) d\mu - (\int_X g(\mathbf{x}) d\mu)^2$. Note that there is no d dependence in the exponent of n. Of course in typical applications, say, in statistical physics, $\mathrm{var}(g)$ can be very large in high dimension. For this reason, one main focus in the field of Monte Carlo methods is variance reduction.

Turning now to the generalization gap,

$$\mathcal{R}(\hat\theta) - \hat{\mathcal{R}}_n(\hat\theta) = I(g) - I_n(g),$$
$$g(\mathbf{x}) = (f(\mathbf{x},\hat\theta) - f^*(\mathbf{x}))^2,$$

where $\hat\theta = \operatorname{argmin} \hat{\mathcal{R}}_n(\theta)$. Note that in this case, the function g is highly correlated with the data; therefore the identity (1) does not apply. One approach for bounding the generalization gap is to estimate the supremum of $I(g) - I_n(g)$ over the hypothesis space. Naturally the scaling of this quantity depends on the hypothesis space. For example, we have:

- for Lipschitz functions (this is related to the Wasserstein distance)

$$\sup_{\|h\|_{Lip} \le 1} |I(h) - I_n(h)| \sim \frac{1}{n^{1/d}},$$

- for functions in the Barron space, defined in [10],

$$\sup_{\|h\|_{\mathcal{B}_1} \le 1} |I(h) - I_n(h)| \sim \frac{1}{\sqrt{n}}.$$

One can see that the curse of dimensionality manifests itself in a different form, namely, the size of the data set.

An important concept for estimating the generalization gap is the Rademacher complexity [3]. Let $\mathcal{H}$ be a set of functions, and let $S = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ be a set of data points. Then, the Rademacher complexity of $\mathcal{H}$ with respect to S is defined as

$$\hat{R}_S(\mathcal{H}) = \frac{1}{n} \mathbb{E}_\xi \left[\sup_{h \in \mathcal{H}} \sum_{i=1}^{n} \xi_i h(\mathbf{x}_i) \right],$$

where $\{\xi_i\}_{i=1}^n$ are i.i.d. random variables taking values ± 1 with equal probability.

For example, we have

- if $\mathcal{H}$ = unit ball in Lipschitz space: $\hat{R}_S(\mathcal{H}) \sim O(1/n^{1/d})$
- if $\mathcal{H}$ = unit ball in C^0, the space of continuous functions: $\hat{R}_S(\mathcal{H}) \sim O(1)$
- if $\mathcal{H}$ = unit ball in the Barron space: $\hat{R}_S(\mathcal{H}) \sim O(1/\sqrt{n})$.

The last one is the optimal scaling that one can expect for the Rademacher complexity of a hypothesis space.

Rademacher complexity is important since it bounds from above and below the supremum value we are interested in [3]: Given a function class $\mathcal{H}$, for any $\delta \in (0,1)$, with probability at least $1-\delta$ over the random samples $S=(\mathbf{x}_1,\dots,\mathbf{x}_n)$, we have

$$\sup_{h\in\mathcal{H}}\left|\mathbb{E}_{\mathbf{x}}\left[h(\mathbf{x})\right]-\frac{1}{n}\sum_{i=1}^{n}h(\mathbf{x}_i)\right| \le 2\hat{R}_S(\mathcal{H})+\sup_{h\in\mathcal{H}}\|h\|_\infty\sqrt{\frac{\log(2/\delta)}{2n}},$$

$$\sup_{h\in\mathcal{H}}\left|\mathbb{E}_{\mathbf{x}}\left[h(\mathbf{x})\right]-\frac{1}{n}\sum_{i=1}^{n}h(\mathbf{x}_i)\right| \ge \frac{1}{2}\hat{R}_S(\mathcal{H})-\sup_{h\in\mathcal{H}}\|h\|_\infty\sqrt{\frac{\log(2/\delta)}{2n}}.$$

With this background, we can now divide all machine learning models into two classes.

1. Models that suffer from the curse of dimensionality:

   $$\text{generalization error} \ge O(m^{-\alpha/d}) \quad \text{or} \quad O(n^{-\beta/d}).$$

   The curse of dimensionality can come from either the approximation error (the first term at the right-hand side) or the generalization gap (the second term at the right-hand side). Piecewise polynomial approximation and wavelets with fixed basis belong to this class.
2. Models that do not suffer from the curse of dimensionality:

$$\text{generalization error} \le O(\gamma_1(f^*)/m+\gamma_2(f^*)/\sqrt{n}).$$

Three machine learning models have been identified in this class:

- random feature models: Let $\{\phi(\cdot,\omega),\omega\in\Omega\}$ be a set of random "features" and denote by π the probability distribution of the random variable ω. Given any i.i.d. realization $\{\omega_j\}_{j=1}^m$ of ω,

$$\mathcal{H}_m(\{\omega_j\})=\{f_m(\mathbf{x},\theta)=\frac{1}{m}\sum_{j=1}^{m}a_j\phi(\mathbf{x};\omega_j)\}.$$

- two-layer neural networks:

$$\mathcal{H}_m=\{\frac{1}{m}\sum_{j=1}^{m}a_j\sigma(\boldsymbol{b}_j^T\mathbf{x}+c_j)\}.$$

- residual neural networks: $\mathcal{H}_m=\{f(\cdot,\theta)=\alpha\cdot \boldsymbol{z}_{L,L}(\cdot)\}$:

$$\boldsymbol{z}_{l+1,L}(\mathbf{x})=\boldsymbol{z}_{l,L}(\mathbf{x})+\frac{1}{L}\boldsymbol{U}_l\sigma\circ(\boldsymbol{W}_l\boldsymbol{z}_{l,L}(\mathbf{x})),$$
$$l=0,\dots,L-1,\qquad \boldsymbol{z}_{0,L}(\mathbf{x})=\boldsymbol{V}\mathbf{x}.$$

  Here L is the depth of the network.

To identify these models, one proceeds in the following steps [10]:

1. Express the target function as an expectation through some law of large numbers. This is straightforward for the random feature model and the two-layer neural network model. For residual network models, this is the "compositional law of large numbers" established in [10].
2. Establish the rate of convergence for the approximation error by proving the corresponding central limit theorem. Again this is standard for the first two models, but requires some work for the residual network model.
3. Estimate the Rademacher complexity. It turns out that for all three models, the Rademacher complexity has the optimal scaling as a function of the size of the dataset.

In this process, one also identifies the right function spaces for the corresponding machine learning model. For the random feature model, this is the corresponding reproducing kernel Hilbert space. For the two-layer neural network and residual network models, this is the Barron space and compositional function space, respectively, defined in [10].

A consequence is that one can readily prove dimension-independent (Monte Carlo-like) a priori error estimates for the generalization error for suitably regularized versions of these models.

Much remains to be done in order to build a solid mathematical foundation for machine learning. However, it is clear that the issues are very much in the spirit of numerical analysis. The new twists are that the dimensionality is high, and the models can be overparametrized in the sense that there are more parameters than data.

Conclusion

What I have touched upon is only the tip of a huge incoming iceberg. We are on the verge of a new scientific revolution that will impact not only science but also mathematics and applied mathematics in fundamental ways. In particular,

- Integrating machine learning (representing the Keplerian paradigm) with first-principle-based physical modeling (representing the Newtonian paradigm) opens up a new and powerful paradigm for scientific research. Applied mathematics is the most natural platform for this integration.
- To build the theoretical foundation of machine learning, one has to develop high-dimensional numerical analysis.

If the evidence for these claims is not yet strong enough, it continues to build at an amazing speed. It is hard to imagine a better opportunity for applied mathematics than what is lying in front of us.

ACKNOWLEDGMENT. Machine learning is an area in which young people are playing a particularly active role. This is also reflected in the work reported here. In particular, I am grateful to my young collaborators for their contribution to the work reported here: Jiequn Han, Arnulf Jentzen, Qianxiao Li, Chao Ma, Zheng Ma, Cheng Tai, Han Wang, Qingcan Wang, Lei Wu, Linfeng Zhang, and Yajun Zhou. I am also grateful to Reza Malek-Madani for his suggestions, which helped to improve the draft.

References

[1] Ainsworth M and Oden JT, *A Posteriori Error Estimation in Finite Element Analysis*, Wiley, 2000. MR1885308

[2] Bardos C, Golse F and Levermore D, Fluid dynamic limits of kinetic equations. I. Formal derivations, *J. Statist. Phys.*, Vol. 63, No. 1/2, 1991. MR1115587

[3] Bartlett PL and Mendelson S, Rademacher and Gaussian complexities: risk bounds and structural results, *J. Mach. Learn. Res.*, 3(Nov):463–482, 2002. MR1984026

[4] Barron AR, Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Trans. Inform. Theory* 39(3):930–945, 1993. MR1237720

[5] Behler J and Parrinello M, Generalized neural-network representation of high-dimensional potential-energy surfaces, *Phys. Rev. Lett.* 98, 146401, 2007.

[6] Cai ZN, Fan YW and Li R, Globally hyperbolic regularization of Grad's moment system in one dimensional space, *Comm. Math. Sci.* 11(2), 2012, pp. 547–571. MR3002565

[7] Dirac PA, Quantum mechanics of many-electron systems, *Proc. Roy. Soc. London*. Series A, Vol. 123, No. 792, 1929.

[8] E W, *Principles of Multiscale Modeling*, Cambridge University Press, 2011. MR2830582

[9] E W, Han J, and Jentzen A, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, *Comm. Math. Stats.*, vol. 5, no. 4, pp. 349–380, 2017. MR3736669

[10] E W, Ma C, and Wu L, Barron spaces and the compositional function spaces for neural network models, arXiv.org/abs/1906.08039, 2019.

[11] E W and Zhou Y, A mathematical model for linguistic universals, arXiv.org/abs/1907.12293, 2019.

[12] Han J, Ma C, Ma Z, and E W, Uniformly accurate machine learning based hydrodynamic models for kinetic equations, arXiv:1907.03937, 2019.

[13] Parr RG and Yang W, *Density-Functional Theory of Atoms and Molecules*, Oxford Science Publications, 1989.

[14] Pardoux E and Peng SG, Adapted solution of a backward stochastic differential equation, *Systems Control Lett.*, Vol. 14, Issue 1, 1990. MR1037747

[15] Van Dyke M, *Perturbation Methods in Fluid Mechanics*, annotated edition, Parabolic Press, 1975. MR0416240

[16] Wang H, Zhang LF, Han J, and E W, DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics, *Comput. Phys. Comm.*, vol. 228, pp. 178–184, 2018. `https://github.com/deepmodeling/deepmd-kit`.

[17] Zhang L, Han J, Wang H, Saidi W, Car R, and E W, End-to-end symmetry preserving inter-atomic potential energy model for finite and extended systems, NIPS, 2018.

[18] Zhang L, Wang H, Car R, and E W, Active learning of uniformly accurate inter-atomic potentials for materials simulation, arXiv:1810.11890, 2018.

Weinan E

Credits

Opening image is courtesy of Getty.
Photo of the author is by Yiming Fu.