Commun. Math. Stat. (2017) 5:1–11
DOI 10.1007/s40304-017-0103-z

Dedicated to Professor Chi-Wang Shu on the occasion of his 60th birthday

A Proposal on Machine Learning via Dynamical Systems

Weinan E[1,2,3]

Received: 7 February 2017 / Revised: 21 February 2017 / Accepted: 24 February 2017 /
Published online: 22 March 2017

Abstract We discuss the idea of using continuous dynamical systems to model general high-dimensional nonlinear functions used in machine learning. We also discuss the connection with deep learning.

Keywords Deep learning · Machine learning · Dynamical systems

Mathematics Subject Classification 37N99

The number one task in machine learning is to efficiently create a sufficiently rich class of functions that can represent the data with the desired accuracy. The most straightforward approach is that of approximation theory: One starts with linear functions and then builds nonlinear ones using splines, wavelets or other basis functions [1]. The obvious obstacle with this approach is the curse of dimensionality. To deal with this issue, one often has to make simplified "ansatz," by postulating special forms for the functions, for example an additive form or multiplicative form [2].

In recent years, a new class of techniques has shown remarkable success, the deep neural network model [3]. Neural network is an old idea, but recent experience has shown that deep networks with many layers seem to do a surprisingly good job in modeling complicated data sets. The difference between neural networks and traditional approximation theory is that neural networks use compositions of simple functions to approximate complicated ones, i.e., the neural network approach is compositional,

✉ Weinan E
weinan@math.princeton.edu

1 Beijing Institute of Big Data Research (BIBDR), Beijing, China

2 Department of Mathematics and PACM, Princeton University, Princeton, NJ, USA

3 Center for Data Science and BICMR, Peking University, Beijing, China

whereas classical approximation theory is usually additive. Although we still lack a theoretical framework for understanding deep neural networks, their practical success has been very encouraging.

In this note, we go one step further, by exploring the possibility of producing nonlinear functions using continuous dynamical systems, pushing the compositional approach to an infinitesimal limit. In the framework of supervised learning, this gives rise to a new class of control problems. In this view, the deep neural networks can be thought of as being discrete dynamical systems. Compared with deep neural networks, there are several potential advantages with a continuous approach.

1. Mathematically it is easier to think about and deal with continuous dynamical systems. Continuous formulation offers more flexibility (for example adding constraints, adapting the dynamical system to the problem, imposing structure on the dynamical system), and it is easier to analyze continuous dynamical systems than discrete ones.
2. Deep neural network can be thought of as a discretization of the continuous dynamical systems. However, from the viewpoint of discretizing dynamical systems, there are many possibilities that one can explore. For example, one can use adaptive time step size, which corresponds to choosing the layers adaptively. One can use high order or even implicit discretization, and these do not yet have an analog in deep neural networks. One can also use advanced numerical methods for training, such as the multi-grid method or the parallel shooting method (see [4,5]).
3. Most models in physical sciences (physics, chemistry, etc) are represented using dynamical systems in the form of differential equations. The continuous dynamical systems approach makes it easier to combining ideas from machine learning and physical modeling.
4. The vast majority of the applied mathematics community is familiar with differential equations. The continuous dynamical systems approach to machine learning will be of particular interest to them.

In this note we only introduce some of the simplest ideas, leaving their further exploration and practical implementation to future work (see for example [4,5]). We note that continuous dynamical systems approach was mentioned in [6–8]. The connection to control theory has been explored in the work of [7], among others. See also the interesting blog by Recht [9]. However, it is the author's believe that there is still lots of room for development in this setting.

1 Formulation

Consider the differential equation in $\mathbb{R}^d$:

$$\frac{\mathrm{d}z}{\mathrm{d}t} = f(t, z), \quad z(0) = x. \tag{1.1}$$

The solution at time t will be denoted by $z(t, x)$. Fix a time horizon T, the *flow map*:

$$x \to z(T, x) \tag{1.2}$$

gives rise to a function of x, which in general is nonlinear. The basic idea behind the dynamical system approach to supervised learning is to tune f so that the flow map can produce the kind of nonlinear function needed to fit the data.

We will consider supervised learning. At the abstract level, the problem can be formulated as follows. Let μ be a probability distribution on a domain D in $\mathbb{R}^d$. Let y be a function on D. If y is a discrete valued function, then the problem is called a classification problem. Otherwise, the problem is called a regression problem. Let $\{x_i\}$ be a finite sample of μ, and $y_i = y(x_i) + \varepsilon_i$ be the label associated with x_i. Here ε_i denotes measurement noise. Our problem is to predict y from the data $\{(x_i, y_i)\}$.

There are several well-known very elementary supervised learning models, for example linear regression for the regression problem and support vector machine (SVM) and logistic regression for the classification problem. Associated with each of these models, there is a loss function. In the case of linear regression, the loss function is $(u(x_i) - y_i)^2$ where u is the fitted linear function. Below we will focus on the regression problem.

Consider the dynamical system:

$$\frac{\mathrm{d}z}{\mathrm{d}t} = f(A(t), z), \quad z(0) = x, \tag{1.3}$$

where f is a function chosen beforehand, A is the control. Fix a time horizon T. Our goal is to choose A such that the flow map at time T can be fitted accurately with linear regression. Let $u_\alpha(x) = \alpha_1 \cdot z(T, x) + \alpha_0$ where $\alpha_1 \in \mathbb{R}^d$, $\alpha_0 \in \mathbb{R}^1$, and $\alpha = (\alpha_0, \alpha_1)$. Our objective is to

$$\min_{\alpha, A} \int_D \|y(x) - u(x)\|^2 \mathrm{d}\mu(x). \tag{1.4}$$

Obviously we can use more general loss function L in the last expression:

$$\min_{\alpha, A} \int_D L(y(x), u(x)) \mathrm{d}\mu(x). \tag{1.5}$$

In practice, we only have a finite set of data $\{(x_i, y_i)\}$, where $\{x_i\}$ is a finite sample of μ and y_i is the label associated with x_i, so (1.3) should really be replaced by:

$$\min_{\alpha, A} \sum_{i=1}^{N} (y_i - u(x_i))^2. \tag{1.6}$$

(1.5) and (1.6) have the flavor of a control problem of an unusual type. Here we are interested in a large collection of initial data, instead of a single initial configuration and a single target.

We need to specify the function f. This is where some creative thinking has to take place. One simplest possible example is:

1. Component-wise nonlinearity

$$f(A(t), z) = (\beta_1(t)F(A_1(t)z_1), \beta_2(t)F(A_2(t)z_2), \cdots, \beta_d(t)F(A_d(t)z_d))^T, \tag{1.7}$$

   where F is a simple nonlinear function, for example the sigmoid function: $F(t) = 1/(1 + e^{-t})$. We have also added the functions $(\beta_1, \beta_2, \cdots, \beta_d)$ to the set of controls.
2. Full nonlinearity

$$f(A(t), z) = \beta(t)F(A(t)z), \tag{1.8}$$

   where F is some simple vector-valued nonlinear function.

It is straightforward to extend this formulation to other supervised learning models such as the logistic regression and support vector machines.

1.1 Finding the Optimal Control

To find the optimal control, we almost always need to compute the gradient of the objective function with respect to the control. Here we show how this can be done. Our notation is a bit sloppy here. But this makes things more transparent.

Fix $t \in [0, T]$. Define

$$I(\alpha, A) = \int (y(x) - u(x))^2 \, \mathrm{d}\mu(x). \tag{1.9}$$

We will compute $\frac{\delta I}{\delta A(t)}$. It is easy to see that

$$\frac{\delta I}{\delta A(t)} = -2\alpha_1 \cdot \int (y(x) - \alpha_1 \cdot z(T, x) - \alpha_0) \frac{\delta z(T, x)}{\delta A(t)} \mathrm{d}\mu(x). \tag{1.10}$$

To compute $\frac{\delta z(T,x)}{\delta A(t)}$, consider the dynamical system in which the control is kept the same except in a small neighborhood around t, A is perturbed by a small amount. The leading order perturbation to the solution of the dynamical system is given by the following problem:

$$\frac{\mathrm{d}v(\tau)}{\mathrm{d}\tau} = \nabla_z f(A(\tau), z(\tau, x))v(\tau), \quad \tau > t \tag{1.11}$$

with the condition $v(t) = \nabla_A f(A(t), z(t, x))$. We then have

$$\frac{\delta z(T, x)}{\delta A(t)} = v(T). \tag{1.12}$$

At the first sight, (1.11) and (1.12) should be solved forward in time. However, note that we are interested in solving this equation for multiple values of t. Therefore, it might be more advantageous to solve (1.11) and (1.12) backward in time, i.e., first by forming the product of the propagators from time T to time t and then multiplying this propagator by the initial value at time t. In this way, (1.11) and (1.12) can be solved simultaneously for all t in one sweep. This corresponds to the idea of back propagation.

With these results in place, it is now clear how to formulate various optimization algorithms, including the gradient decent and stochastic gradient decent algorithm.

2 Connection with Deep Neural Network

Deep neural networks can be considered as discrete dynamical systems, the basic dynamics at each step being a linear transformation followed by a component-wise nonlinear (activation) function. As such, it is among the simplest nonlinear discrete dynamical systems one can imagine. These discrete dynamical systems offer another feature, namely the dimensionality of the system can be changed from step to step, simply by changing the size of the linear transformation.

The continuous dynamical systems approach advocated here can in principle incorporate more general dynamics. This added flexibility can be important in many situations, such as the ones encountered in physical science. For example, when trying to learn the inter-atomic potential of a molecular dynamics system with both short and long range interactions, this added flexibility should come in handy. Continuous dynamical systems do not allow the change of dimensionality. Therefore, the lifting and projection of dynamics to higher and lower dimensions have to be performed at the beginning or end of the dynamics, i.e., at $t = 0$ or $t = T$.

The behavior of dynamical systems at large times is a notoriously difficult problem in mathematics, particularly for discrete dynamical systems. One may encounter situations where the dynamics explodes, converges to stationary states or exhibits chaotic behavior. This may explain some of the difficulties (such as the vanishing and explosion of gradients) encountered in the training of deep neural networks. Continuous dynamical systems also experience these difficulties. However, in the continuous setting, several ideas can be explored to help alleviating these problems:

1. Imposing structure on the dynamical system. The simplest structures that one can imagine are the Hamiltonian structure or the gradient flow structure [10].
2. Sophisticated numerical discretization methods, such as adaptive time stepping, implicit or semi-implicit algorithms. These are particularly helpful for long time (large time horizon) computations.

2.1 Connection with Deep Residual Networks

To better appreciate these points, we take a more detailed look at the deep residual networks. In its general form, deep residual networks are described by the following discrete dynamical system [11]:

$$y_l = h(z_l) + \mathscr{F}(z_l, W_l), \tag{2.1}$$

$$z_{l+1} = g(y_l), \tag{2.2}$$

where z_l and z_{l+1} are the input and output of the l-th layer, y_l is an auxiliary variable for the l-th layer, h and g are some mappings which could in principle be nonlinear. A main result of [11], found through numerous numerical experiments, is that for very deep networks (hundreds or thousands of layers), training is the easiest if both g and h are the identity map [11]. This is quite expected from the viewpoint of dynamical systems. In fact, denote by G the inverse map of g, we can then write the above dynamical system as:

$$z_{l+1} = G(h(z_l) + \mathscr{F}(z_l, W_l)). \tag{2.3}$$

In order to have a stable behavior (nonvanishing or exploding), the gradient of the right hand side should be close to an identity map. Assuming that $\mathscr{F}$ is a small perturbation, we then need

$$\nabla G \nabla h \sim I. \tag{2.4}$$

This is certainly fulfilled when both g and h are identity maps. In general, one has

$$z_{l+1} \sim G(h(z_l)) + \nabla G \cdot \mathscr{F}(z_l, W_l). \tag{2.5}$$

This means that the leading order behavior is dominated by $G(h)$. The added flexibility from choosing more general g and h does not offer much real improvement if one wants to use a large number of layers.

Let g and h both be identity maps. Then (2.3) becomes:

$$z_{l+1} = z_l + \mathscr{F}(z_l, W_l). \tag{2.6}$$

This can be viewed as a discretization of the dynamical system:

$$\frac{\mathrm{d}z}{\mathrm{d}t} = \mathscr{F}(z, W(t)). \tag{2.7}$$

In fact the simplest discretization of (2.7) takes the form:

$$z_{l+1} = z_l + \Delta t_l \mathscr{F}(z_l, W_l), \tag{2.8}$$

where Δt_l is the step size at the l-th time step. By adaptively choosing Δt_l, one can improve the efficiency and stability of the algorithm at the same time [12]. In the language of deep learning, adaptive time stepping means adaptively choosing the layers.

Remark It should be noted that contrary to the conventional problem of solving differential equations, here our concern is not the dynamical system itself, but rather the representation of the data. Therefore, accuracy in the numerical approximation of

the differential equation is not our primary concern, the training error, efficiency and stability in the training are our main interests.

3 The Representability and Controllability Problem

One important question is whether the flow map of a given dynamical system can represent the data at hand. The same issue is also baffling the deep learning community. From our perspective, this representability problem can be viewed as a controllability problem.

The standard controllability problem asks the question: Given an initial and a target position, can we use control to move the system from the initial position to an arbitrarily small neighborhood of the target position during the specified time horizon. For some general results, we refer to [13]. Our concern here is different. Here we are interested in the following question. Given a final supervised learning model, such as linear regression, can we use the control to move the flow map to a set in which the specified model works satisfactorily?

To get some clear idea about this, let us consider an idealized version of this problem: Given a mapping M on D and given T, can we choose the control such that $z(x, T) = M(x)$ for all $x \in D$?

The one-dimensional version of this problem may shed some light. Consider the case when our dynamical system takes the form:

$$\frac{\mathrm{d}z}{\mathrm{d}t} = A(t) f(z) \tag{3.1}$$

For simplicity of illustration, we assume that f does not vanish. We can rewrite the above equation as

$$\frac{1}{f(z)} \frac{\mathrm{d}z}{\mathrm{d}t} = A(t) \tag{3.2}$$

Integrating both sides, we obtain

$$G(M(x)) - G(x) = \int_0^T A(t)\mathrm{d}t \tag{3.3}$$

where G is the primitive function of $1/f$. The right hand side of (3.3) is independent of x. Therefore for (3.3) to hold, the left hand side should also be independent of x. This imposes a severe constraint on M and f.

This example is highly simplified. But it gives us some hints about the issues involved in understanding whether the dynamical system approach gives a good model to the learning problem at hand. In general, the viewpoint of exact representability is very limited. Instead, one should take an approximation viewpoint. For that purpose, one may consider lifting the dynamical system to higher and higher dimensions, i.e., instead of consider the dynamical system in the same dimension as the data, one may consider the dynamical system in a different dimension, and use

$$z(0) = Px \tag{3.4}$$

as the initial condition. Here P is a lifting matrix. One may also consider extending (1.8) to

$$f(A(t), z) = \sum_{k=1}^{n} \beta_k(t) F_k(A_k(t)z). \tag{3.5}$$

In the general case, if the mapping M is smoothly homotopic to the identity map, i.e., if there exists a one parameter family of mappings $G(\cdot) : [0, 1] \to \text{diff}(D)$, where $\text{diff}(D)$ is the space of all diffeomorphism on D, such that $G(0, x) = x$, $G(1, x) = M(x)$, then there always exists an $f : [0, 1] \times D \to \mathbb{R}^d$, such that the flow map generated by the dynamical system

$$\frac{\mathrm{d}z}{\mathrm{d}t} = f(t, z)$$

gives rise to M: $z(1, x) = M(x)$.

To see this one just has to define $f(t, z) = \frac{\partial G}{\partial t}(t, G^{-1}(z))$, where G^{-1} is the inverse of the mapping G at time t.

4 Extensions

4.1 Continuum in Space

For some tasks in computer vision and artificial intelligence, it is of interest to view the data itself as being continuum objects. This situation can be found, for example, in image recognition and voice recognition. In this case, we need to introduce another independent variable to represent space.

When the spatial variable is taken into account, the most commonly seen dynamical systems that are in the form of partial differential equations (PDEs). These dynamical systems are local since differentiation is a local operation. For applications in vision, the situation seems to be quite different. The dynamical systems that are of interest should be nonlocal in nature.

One way of introducing nonlocal operation is to use integral operators. For example, we may replace $A(t)z$ in (1.7) by $\int K(x, y, t)z(y, t)\mathrm{d}y$ where K is some kernel function (in this section we use x and y to denote the spatial variable). To ensure translation invariance, the kernel function should take the form of a convolution:

$$\int K(x, y, t)z(y, t)\mathrm{d}y = \int K_0(x - y, t)z(y, t)\mathrm{d}y = (K_0 * z)(x, t). \tag{4.1}$$

The simplest dynamical system that uses this convolution operator may take the form:

$$\frac{\partial z}{\partial t}(x, t) = f((K_0 * z)(x, t)), \tag{4.2}$$

where f is some nonlinear function. This leads to a model that is similar in spirit to the convolutional neuron network model.

4.2 Adding Constraints

The dynamical system formulation also allows us to impose constraints with relative ease. For example, one might ask the control A to be orthogonal at each time step: $A(t)^T A(t) = I$ where I is the identity matrix. One might also impose constraints on z. This can be done by introducing suitable Lagrange multipliers.

4.3 Structure of the Dynamical System

We may ask the dynamical system to have certain structure. For example we may ask the dynamical system to be a gradient flow. One such example is given below (see the section on density estimation). We may also ask the dynamical system to have a Hamiltonian structure. This direction remains to be explored.

4.4 Regularization

Within the context of control theory, it is natural to consider regularized models, by adding regularization terms on the control. For example, two natural choice of regularization terms are $\int_0^T \|A(t)\|_1 \mathrm{d}t$ and $\int_0^T \|A(t)\|_F^2 \mathrm{d}t$ where $\|A(t)\|_1$ and $\|A(t)\|_F$ are the L^1 norm and the Frobenius norm of $A(t)$, respectively. In the language of control theory, these regularization terms give rise to the "running cost" for the model (the costs considered in (1.4), (1.5), (1.6) are "terminal costs"). The corresponding Bellman equation becomes less degenerate in the presence of running costs.

Deep neural network models usually do not include such regularization terms, except that pooling and drop out operations often used there may lead to some regularization effects.

5 Other Examples

There are numerous other examples of learning via dynamical systems. Here we only mention a few.

5.1 Clustering Dynamics

One well-known model of clustering dynamics is the Hegselmann–Krause System. An interesting noisy form of this dynamics is described by the following system of stochastic differential equations (see [14]):

$$\mathrm{d}x_i = -\frac{1}{N} \sum_{j,\, |x_i - x_j| \le R} (x_i - x_j) + \sigma \mathrm{d}W_i, \quad x_i(0) = x_i^0. \tag{5.1}$$

Here the $\{x_i\}$'s are the interacting agents, the $\{W_i\}$'s are independent Wiener processes, R is the interaction range. This can be considered as a clustering algorithm for the data set $\{x_i^0\}$. In the continuous limit when there is a continuum of agents, the density of the agents ρ evolves according to:

$$\frac{\partial \rho(x,t)}{\partial t} = -\rho(x,t)\int_{|x-y|\le R}(x-y)\rho(y,t)\mathrm{d}y + \frac{\sigma^2}{2}\Delta\rho(x,t). \tag{5.2}$$

As shown in [14], PDE (partial differential equations) methods can be quite powerful in helping us to understand the dynamics of the Hegselmann–Krause System.

5.2 Density Estimation

In a very interesting paper, Tabak and Vanden-Eijnden proposed performing density estimation by mapping the initial distribution of data to a target distribution, usually an isotropic Gaussian [15]. This mapping can be obtained from the gradient flow dynamics of the log-likelihood function. In the continuous form (when the data set is a continuum), the flow dynamics takes the form:

$$\frac{\partial \phi(x,t)}{\partial t} = \frac{\nabla\mu(x)}{\mu(x)}\rho(x,t) - \nabla\rho(x,t), \quad \rho(x,t) = \frac{\rho_0(z)}{J(z,t)}, \quad x = \phi(z,t). \tag{5.3}$$

Here ϕ is the flow map, ρ_0 is the initial distribution, μ is the target distribution (usually a Gaussian), J is the Jacobian of the flow map. The evolution of the density takes a simpler form:

$$\frac{\partial \rho(x,t)}{\partial t} = \nabla\left(\left(\nabla\rho(x,t) - \frac{\nabla\mu(x)}{\mu(x)}\rho(x,t)\right)\rho(x,t)\right). \tag{5.4}$$

The idea behind this can be useful for other unsupervised learning tasks.

6 Conclusions

We proposed some ideas about using continuous dynamical systems as a tool for machine learning. Besides offering some algorithmic and analytical advantages, we also hope that this provides an attractive alternative view to deep learning.

There are numerous open questions. The most important one is whether there are other natural candidates for the underlying dynamical systems besides the ones suggested in deep learning. Practical issues such as discretization and advanced numerical algorithms also remain to be explored.

Acknowledgements This is part of an ongoing project with several collaborators, including Jiequn Han, Qianxiao Li, Jianfeng Lu and Cheng Tai. The author benefitted a great deal from discussions with them, particularly Jiequn Han. This work is supported in part by the Major Program of NNSFC under Grant 91130005, ONR N00014-13-1-0338 and DOE DE-SC0009248.

References

1. Fan, J., Gijbels, I.: Local Polynomial Modeling and Its Applications. Chapman & Hall, London (1996)
2. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction a, Springer Series in Statistics, second edition, (2013)
3. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
4. Han, J., E, W.: in preparation
5. Li, Q., Tai, C., E, W.: in preparation
6. Almeida, L.B.: A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In: Proceedings ICNN 87. San Diego, IEEE (1987)
7. LeCun, Y.: A theoretical framework for back propagation. In: Touretzky, D., Hinton, G., Sejnouski, T. (eds.) Proceedings of the 1988 connectionist models summer school, Carnegie-Mellon University, Morgan Kaufmann, (1989)
8. Pineda, F.J.: Generalization of back propagation to recurrent and higher order neural networks. In: Proceedings of IEEE conference on neural information processing systems, Denver, November, IEEE (1987)
9. Recht, B.: http://www.argmin.net/2016/05/18/mates-of-costate/
10. E, W., Ming, P.: Calculus of Variations and Differential Equations, lecture notes, to appear
11. He, K., Zhang, X., Ren, S., Sun, J.: Identity mapping in deep residual networks. (July, 2016) arXiv:1603.05027v3
12. Lambert, J.D.: Numerical Methods for Ordinary Differential Systems: The Initial Value Problem. Wiley, New York (1992)
13. Stroock, D.W., Varadhan, S.R.S.: Multi-Dimensional Diffusion Processes. Springer, Berlin (2006)
14. Wang, C., Li, Q., E, W., Chazelle, B.: Noisy Hegselmann–Krause systems: phase transition and the 2R-conjecture. In: Proceedings of 55th IEEE Conference on Decision and Control, Las Vegas, (2016) (Full paper at arXiv:1511.02975v3, 2015)
15. Tabak, E.G., Vanden-Eijnden, E.: Density estimation by dual ascent of the log-likelihood. Commun. Math. Sci. **8**(1), 217–233 (2010)