

Machine Learning, Deepest Learning: Statistical Data Assimilation Problems

Henry Abarbanel,
Department of Physics
and

Marine Physical Laboratory (Scripps Institution of Oceanography)
Center for Engineered Natural Intelligence
habarbanel@ucsd.edu

Paul Rozdeba

and

Sasha Shirman

Department of Physics
University of California, San Diego
9500 Gilman Drive, Mail Code 0374
La Jolla, CA 92093-0374, USA

Draft of July 6, 2017

1 Abstract

We formulate a strong equivalence between machine learning, artificial intelligence methods and the formulation of statistical data assimilation as used widely in physical and biological sciences. The correspondence is that layer number in the artificial network setting is the analog of time in the data assimilation setting. Within the discussion of this equivalence we show that adding more layers (making the network deeper) is analogous to adding temporal resolution in a data assimilation framework.

How one can find a candidate for the global minimum of the cost functions in the machine learning context using a method from data assimilation is discussed. Calculations on simple models from each side of the equivalence are reported.

Also discussed is a framework in which the time or layer label is taken to be continuous, providing a differential equation, the Euler-Lagrange equation, which shows that the problem being solved is a two point boundary value problem familiar in the discussion of variational methods. The use of continuous layers is denoted “deepest learning”. These problems respect a symplectic symmetry in continuous time/layer phase space. Both Lagrangian versions and Hamiltonian versions of these problems are presented. Their well-studied implementation in a discrete time/layer, while respected the symplectic structure, is addressed. The Hamiltonian version provides a direct rationale for back propagation as a solution method for the canonical momentum.

2 Introduction

Through the use of enhanced computational capability two, seemingly unrelated, ‘inverse’ problems have flourished over the past decade. One is machine learning in the realm of artificial intelligence [1, 12, 21] with developments that often go under the name “deep learning”. The other is data assimilation in the physical and life sciences. This describes the transfer of information from observations to models of the processes producing those observations [4, 9, 2].

This paper is directed towards demonstrating that these two areas of investigation are the same at a fundamental level. Each is a statistical physics problem where methods utilized in one may prove valuable for the other.

The main goal of the paper is to point out that many developments in data assimilation can be utilized in the arena of machine learning. We also suggest that innovative methods from machine learning may be valuable in data assimilation.

Two areas of focus are attended to here: (1) a variational annealing (VA) method for the action (cost function) for machine learning or statistical data assimilation that permits the location of the apparent global minimum of that cost function. (2) The notion of analyzing each problem in continuous time or layer, which we call **deepest learning**, wherein it is clear that one is addressing a two point boundary value problem [33, 11] with an underlying symplectic structure. Methods abound for solving such two point boundary value problems [27] and for assuring that symplectic structures are respected when time (or layer) is discretized. These may be quite fruitful in machine learning.

This paper primarily discusses multilayer perceptrons or feedforward networks [12] though it also makes it is clear that the discussion carries over to recurrent networks as well [14, 8, 26].

3 Background

3.1 Machine Learning; Standard Feedforward Neural Nets

We begin with a brief characterization of simple architectures in feedforward neural networks [1, 12, 21]. The network we describe is composed of an input layer l_0 and output layer l_F and hidden layers $l_1, l_2, \dots, l_F - 1$. Within each layer we have N active units, called ‘neurons’, each of which has d degrees-of-freedom. (Usually d is chosen to be 1.)

Each layer has N neurons with d degrees of freedom, so each layer has $D = Nd$ degrees-of-freedom. For our purposes the ‘neurons’ in each layer are the same structure. This can be generalized to different numbers and different types of neurons in each layer at the cost of a notation explosion.

Data is available to layer l_0 and layer l_F in M pairs of L -dimensional input, at l_0 , and output, at l_F . These are sets of vectors: $\{y_r^{(k)}(l_0), y_r^{(k)}(l_F)\}$ where $k = 1, 2, \dots, M$ labels the pairs, r is an index on the L -dimensional data $r = 1, 2, 3, \dots, L \leq D$.

The activity of the units in each hidden layer l , $x_\alpha^{(k)}(l)$; $l_0 < l < l_F$, is

determined by the activity in the previous layer. The index α combines the neuron number j and the neuron degrees-of-freedom a into one label: $\alpha = 1, 2, \dots, Nd = D; j = 1, 2, \dots, N; a = 1, 2, \dots, d$. This connection is described by the nonlinear function $f_\alpha(\bullet)$ via

$$x_\alpha^{(k)}(l) = f_\alpha(\mathbf{x}(l-1), l) = f_\alpha\left(\sum_{\beta=1}^{Nd=D} W_{\alpha\beta}(l)x_\beta^{(k)}(l-1)\right), \quad (1)$$

where $\mathbf{x}^{(k)}(l) = \{x_\alpha^{(k)}(l)\} = \{x_1^{(k)}(l), x_2^{(k)}(l), \dots, x_{Nd}^{(k)}(l)\}$. The summation over weights $W_{\alpha\beta}(l)$ determines how the activities in layer $l-1$ are combined before allowing $f_\alpha(\bullet)$ to act, yielding the activities at layer l . There are numerous choices for the manner in which the weight functions act as well as numerous choices for the nonlinear functions, and we direct the reader to the references for the discussion of the virtues of those choices [1, 12, 21].

At the input and the output layers l_0, l_F the network activities are compared to the observations, and the network performance is assessed using an error metric, often a least squares criterion, or cost function

$$C_M = \frac{1}{M} \sum_{k=1}^M \left\{ \frac{1}{2L} \sum_{r=1}^L R_m(r) \left([x_r^{(k)}(l_0) - y_r^{(k)}(l_0)]^2 + [x_r^{(k)}(l_F) - y_r^{(k)}(l_F)]^2 \right) \right\}, \quad (2)$$

where $R_m(r)$ permits an inhomogeneous weighting in the comparison of the network activities and the data. Minimization of this cost function over all $x_\alpha^{(k)}(l)$ and weights $W_{\alpha\beta}(l)$, *subject to* the network model Eq. (1), is used to determine the weights, the variables $x_\alpha^{(k)}(l)$ in all layers, and any other parameters appearing in the architecture of the network.

Before moving along to data assimilation, we note that one wishes to find the global minimum of the cost function Eq. (2), which is a nonlinear function of the neuron activities, the weights and any other parameters in the functions at each layer. This is an NP-complete problem [25] and as such suggests one cannot find a global minimum of the machine learning problem, as set, unless there is a special circumstance. We will see just such a circumstance in a data assimilation problem equivalent to machine learning tasks.

The machine learning problem as described here assumes there is no error in the model itself, so that the minimization of the cost function Eq. (2) is subject to strong equality constraints through the model. This results in the output at layer l_F $\mathbf{x}^{(k)}(l_F)$ being a very complicated function of the parameters in the model and the activities at layers $\mathbf{x}^{(k)}(l \leq l_F)$. This is likely

connected with the many local minima associated with the NP-complete nature of the search problem.

We introduce a variational annealing (VA) method in the next section which regularizes this by moving the equality constraint into the cost function via a penalty function. This introduces a hyperparameter allowing us to systematically vary the complexity of the search process.

3.2 Standard Statistical Data Assimilation

Now we describe the formulation of a statistical data assimilation problem.

In data assimilation observations are made of a sparse set of dynamical variables, associated with a model of the processes producing the observations. This will allow the estimation of parameters in the model and of the unobserved state variables of the model. The goal is to estimate any unknown parameters in the model, and because not all of the dynamical state variables in the model may be observed, to estimate those unmeasured state variables as well. After a certain time window in which information is transferred to the model, we have an estimate of the full model including an initial condition for all state variables, and predictions are made with the completed model and compared to new observations. This validation by prediction is essentially the same as the question of generalization as addressed in machine learning [12].

In data assimilation, one has a window in time $[t_0, t_F]$ during which observations are made at times $t = \{\tau_1, \tau_2, \dots, \tau_F\}$ which lie in $[t_0 \leq \tau_s \leq t_F]; s = 1, 2, \dots, F$. At each observation time L measurements $y_l(\tau_s); l = 1, 2, \dots, L$ are made, $L \leq D$. Through knowledge of the measurement instruments the observations are related to the state variables of the model via ‘measurement functions’ $h_l(\mathbf{x}) : y_l(\tau_k) = h_l(\mathbf{x}(\tau_k)); l = 1, 2, \dots, L$.

Using what knowledge we have of the processes producing the observations, we develop a dynamical model for the state variables. These satisfy a set of D dynamical differential equations

$$\frac{dx_a(t)}{dt} = F_a(\mathbf{x}(t), t); a = 1, 2, \dots, D. \quad (3)$$

The time dependence of the vector field $\mathbf{F}(\mathbf{x}, t)$ may come from external forcing functions driving the dynamics.

This set of differential equations will necessarily be represented in discrete time when solving them numerically, resulting in a map $x_a(t_k) \rightarrow x_a(t_{k+1}) =$

$f_a(\mathbf{x}(t_k), t_k)$ in which the discrete time vector field $f_a(\bullet)$ is related to $F_a(\mathbf{x}, t)$ via the integration procedure one chooses for Eq. (3).

Starting from an initial condition at t_0 , $x_a(t_0)$ we use the discrete time model

$$x_a(t_{k+1}) = f_a(\mathbf{x}(t_k), t_k) \quad (4)$$

to move forward to the first observation time τ_1 , then to τ_2, \dots eventually reaching the end of the observation window at t_F . Altogether by making N model integration steps in each of the intervals $[\tau_n, \tau_{n+1}]$ we make $(F+1)N$ time steps.: $t_0 \rightarrow \tau_1 \rightarrow \tau_2 \dots \rightarrow \tau_F \rightarrow t_F$.

The measurements are noisy, and the models have errors; so this is a statistical problem. Our goal is to construct the conditional probability distribution, $P(\mathbf{X}|\mathbf{Y})$, of the model states $\mathbf{X}(t_F) = \{\mathbf{x}(t_0), \mathbf{x}(t_1), \dots, \mathbf{x}(t_N), \dots, \mathbf{x}(t_F)\}$, conditioned on the LF measurements $\mathbf{Y}(\tau_F) = \{\mathbf{y}(\tau_1), \mathbf{y}(\tau_2), \dots, \mathbf{y}(\tau_k), \dots, \mathbf{y}(\tau_F)\}$.

Assuming the transition to the model state at time t_{k+1} depends only on the model state at time t_k (that is, the dynamics in Eq. (4) is Markov) and using identities on conditional probabilities [2], one may write the action $A(\mathbf{X}) = -\log[P(\mathbf{X}|\mathbf{Y})]$ (suppressing the dependence on the observations \mathbf{Y} in $A(\mathbf{X})$) as

$$\begin{aligned} A(\mathbf{X}) = & - \sum_{n=1}^F CMI[\mathbf{X}(\tau_n), \mathbf{y}(\tau_n) | \mathbf{Y}(\tau_{n-1})] \\ & - \sum_{n=0}^{N(F+1)-1} \log[P(\mathbf{x}(t_{n+1}) | \mathbf{x}(t_n))] - \log[P(\mathbf{x}(t_0))], \end{aligned} \quad (5)$$

where the conditional mutual information is given as [10] $CMI(a, b|c) = \log \left[\frac{P(a,b|c)}{P(a|c) P(b|c)} \right]$. If the model is error free, $P(\mathbf{x}(t_{n+1}) | \mathbf{x}(t_n))$ is a delta function: $P(\mathbf{x}(t_{n+1}) | \mathbf{x}(t_n)) = \delta^D(\mathbf{x}(t_{n+1}) - \mathbf{f}(\mathbf{x}(t_n), t_n))$.

With a representation of $P(\mathbf{X}|\mathbf{Y})$ we may evaluate conditional expected values of functions $G(\mathbf{X})$ on the path $\mathbf{X}(NF)$ of the model through the observation window $[t_0, t_F]$ as

$$E[G(\mathbf{X})|\mathbf{Y}] = \langle G(\mathbf{X}) \rangle = \frac{\int d\mathbf{X} G(\mathbf{X}) \exp[-A(\mathbf{X})]}{\int d\mathbf{X} \exp[-A(\mathbf{X})]}, \quad (6)$$

in which

$$A(\mathbf{X}) = - \sum_{n=1}^F \log[P(\mathbf{y}(\tau_n) | \mathbf{X}(\tau_n), \mathbf{Y}(\tau_{n-1})]$$

$$- \sum_{n=0}^{N(F+1)-1} \log P[(\mathbf{x}(t_{n+1})|\mathbf{x}(t_n))] - \log[P(\mathbf{x}(t_0))], \quad (7)$$

and terms depending only on the observations were canceled between numerator and denominator in the expected value.

If the observations at the times τ_k are independent, and if the measurement function is the identity $y_r(\tau_k) = h_r(\mathbf{x}(\tau_k)) = x_r(\tau_k)$, and if the noise in the measurements is Gaussian, with a diagonal inverse covariance matrix $R_m(r, \tau_k)$, the first term in the action $A(\mathbf{X})$, the **measurement error term**, takes the form

$$\sum_{n=1}^F \sum_{r=1}^L \frac{R_m(r, \tau_n)}{2} \left(x_r(\tau_n) - y_r(\tau_n) \right)^2. \quad (8)$$

If no measurement is made at τ_k , $R_m(\mathbf{r}, \tau_k) = 0$.

If the error in the model Eq. (4) is taken as additive and Gaussian with diagonal inverse covariance matrix $R_f(a)$ the second term in $A(\mathbf{X})$, the **model error term**, becomes

$$\sum_{n=0}^{N(F+1)-1} \sum_{a=1}^D \frac{R_f(a)}{2} \left(x_a(t_{n+1}) - f_a(\mathbf{x}(t_n), t_n) \right)^2. \quad (9)$$

In each term constants having to do with normalizations of the Gaussians cancel in the expected value. If we keep these constants and take the limit $R_f(a) \rightarrow \infty$, we would restore the delta function in the dynamics.

Finally, if one accepts ignorance of the distribution of initial conditions $P(\mathbf{x}(t_0))$ and selects it as uniform over the dynamical range of the model variables, $\langle G(\mathbf{X}) \rangle$ is evaluated with

$$A_0(\mathbf{X}) = \sum_{n=1}^F \sum_{r=1}^L \frac{R_m(r, \tau_n)}{2} \left(x_r(\tau_n) - y_r(\tau_n) \right)^2 + \sum_{n=0}^{N(F+1)-1} \sum_{a=1}^D \frac{R_f(a)}{2} \left(x_a(t_{n+1}) - f_a(\mathbf{x}(t_n), t_n) \right)^2. \quad (10)$$

$$\langle G(\mathbf{X}) \rangle = \frac{\int d\mathbf{X} G(\mathbf{X}) \exp[-A_0(\mathbf{X})]}{\int d\mathbf{X} \exp[-A_0(\mathbf{X})]}, \quad (11)$$

This is the desired connection between the machine learning formulation (with model error) and the statistical data assimilation formulation: identify layer labels as time $l \Leftrightarrow t$.

This alone could be of passing interest. However, there is much in the connection that may be of some utility. We will discuss these items in the data assimilation language, but the translation should be easy at this point. For statistical data assimilation we call this $A_0(\mathbf{X})$ the standard model.

The critical suggestion here relative to standard practice in machine learning [1, 12, 21], is that by allowing R_f to be finite from the outset—so acknowledging model error—we may add an additional tool for exploration in machine learning environments where typically no account for model error is introduced. Further, the hyper-parameter R_f serves as a regulating device for the complexity of the surface in path space in which the estimation of states and parameters occurs.

4 Data Assimilation Developments of Use in Machine Learning

4.1 Finding the Global Minimum

The key to establishing estimates for unobserved state variables ($L < D$) and for unknown parameters in the model is to perform, approximately of course, the integral Eq. (11). One can do this using Monte Carlo methods [18] or by the method of Laplace [19, 20]. In the Laplace method one seeks minima of the action $A_0(\mathbf{X})$, Eq. (10). The integral is not Gaussian. If it were, we would just do it. As the functions $f_a(\bullet)$ are nonlinear, we must perform a numerical evaluation of Eq. (6).

The Laplace method approximates the integral with contributions from the lowest minima of the action, if one can find them. Minima associated with paths \mathbf{X} having larger action give exponentially smaller contributions to expected values, Eq. (11), than paths with smaller action. This allows one to circumvent a search for the global minimum if the parameters, hyper-parameters [12], and other aspects of the model and data yield a set of action levels connected with minima of the action such that one path yields an action level much smaller than any other path. For numerical accuracy one may use that smallest minimum path (comprised of parameters and ‘hidden’ (unobserved) state variables) neglecting larger minima of the action.

We have developed a variational annealing (VA) approach [32, 33], to finding the path with the smallest value of the action. While we have no proof that the global minimum is found, our numerical results indicate this

may be the case. The VA method produces a set of minima of the action giving a numerical clue as to the roughness of the surface in path \mathbf{X} space.

In data assimilation the surface depends, among other items, on the number of measurements L at each observation time τ_k , on the hyper-parameter R_f , and on the number of model time steps between measurement times τ_n . This translates directly to the analogous machine learning problem with time \rightarrow layer. As the number of model time steps between measurement times increases, the number of hidden layers increases and the model architecture deepens.

VA proceeds by a kind of numerical continuation [3] in R_f of the requirement that varying over all \mathbf{X} and all parameters in $A_0(\mathbf{X})$ minimizes $A_0(\mathbf{X})$. The procedure begins by taking $R_f \rightarrow 0$, namely the complete opposite of the value found in usual machine learning where $R_f \rightarrow \infty$ (deterministic, error free layer to layer maps) from the outset. In the $R_f = 0$ limit, the action is just a quadratic function of the model variables $\mathbf{x}(\tau_k)$ at the times measurements are made, and the minimization is simple: $x_r(\tau_k) = y_r(\tau_k)$ for the $r = 1, 2, \dots, L \leq D$ data presented at the input and output layers. The minimum can be degenerate as we know only $L \leq D$ values for the state variables.

At the first step of VA we choose as a solution to the optimization problem $x_r(\tau_k) = y_r(\tau_k)$ and select the other $D - L$ states as drawn from a uniform distribution with ranges known from the dynamical range of the state variables. One can learn that well enough by solving the underlying model forward for various initial conditions. We make this draw K times, and now have K paths \mathbf{X}^0 as candidates for the VA procedure.

Now we select a small value for R_f , call it R_{f0} . With the previous K paths \mathbf{X}^0 as K initial choices in our minimization algorithm, we find K paths \mathbf{X}^1 for the minimization problem with $R_f = R_{f0}$. This gives us K values of the action $A_0(\mathbf{X}^1)$ associated with the new paths \mathbf{X}^1 .

Next we increase the value of R_f to $R_f = R_{f0}\alpha$ where $\alpha > 1$. (We have found values of α in the range 1.1 to 2 to be good choices). For this new value of R_f , we perform the minimization of the action starting with the K initial paths \mathbf{X}^1 from the previous step to arrive at K new paths \mathbf{X}^2 . Evaluating the action on these paths $A_0(\mathbf{X}^2)$ now gives us an ordered set of actions that are no longer as degenerate. Many of the paths \mathbf{X}^2 may give the same numerical value of the action, however, typically the ‘degeneracy’ lies within the noise level of the data $\approx (1/\sqrt{R_m})$.

This procedure is continued until R_f is ‘large enough’ which is indicated

by at least one of the action levels becoming substantially independent of R_f . As a check on the calculation, we observe that if the action $A_0(\mathbf{X})$ is independent of R_f , its expected value is that of the measurement error term. As the measurement errors were taken to be Gaussian, this term in the action is distributed as χ^2 , and its expected value is readily evaluated. If the action levels are at this expected value of χ^2 for large R_f , the procedure is consistent and no further increases in R_f are required.

Effectively VA starts with a problem ($R_f = 0$) where the global minimum is apparent and systematically tracks it and many other paths through increases in R_f . In doing the ‘tracking’ of the global minimum, one must check that the selected value of α is not too large lest one leave the global minimum and land in another minimum. Checking the result using smaller α is worthwhile.

It is important to note that simply starting with a large value of R_f places one in the undesirable situation of the action $A_0(\mathbf{X})$ having multiple local minima into which any optimization procedure is quite likely to fall.

In the dynamical problems we have examined, one typically finds that as the number of measurements L at each τ_k is increased, fewer and fewer minima of the action remain and when L is large enough there is one minimum. This we attribute to the additional information from the augmented set of measurements, and this will be manifest in the discussion below where the additional information effectively controls unstable directions in the phase space.

4.2 Smallest Minimum; Not Necessarily a Convex Action

As our goal is to provide accurate estimations of the conditional expected value of functions $G(\mathbf{X})$ where \mathbf{X} , a path in model space, is distributed as $\exp[-A(\mathbf{X})]$, we actually do not require convexity of $A(\mathbf{X})$ as a function in path space. From the point of view of accurately estimating expected values, it is sufficient that the lowest action level be **much** smaller than the second lowest action level. If the action value at the lowest level $A(\mathbf{X}_{\text{lowest}})$ is much smaller than the action value at the next minimum $A(\mathbf{X}_{\text{second lowest}})$, then by a factor $\exp[-\{A(\mathbf{X}_{\text{lowest}}) - A(\mathbf{X}_{\text{second lowest}})\}]$, the lowest path $\mathbf{X}_{\text{lowest}}$ dominates the integral to be done and provides a sensible choice for the path at which to evaluate the integral.

5 Examples from Feedforward Neural Networks and from Data Assimilation

In this section we examine one example from multi-layer perceptrons and one example from statistical data assimilation. The latter utilizes a differential equation model introduced by Lorenz in 1996 [23] which permits one to easily increase the number of dimensions of the phase space, to easily select the number of observations within a designated measurement window, and to easily choose the number of model evaluations between measurement times. The latter is analogous to increasing the number of layers in a multi-layer perceptron.

In each case we perform a ‘twin experiment’. We use a model to generate solutions from some initial conditions. These solutions, when Gaussian noise is added to them, become our noisy data. Using the noisy data we use VA to estimate the unobserved state variables (hidden layer variables) and parameters/weights.

5.1 Data Assimilation for Lorenz96 Model

We begin by examining the dynamical equations introduced by [23]:

$$\frac{dx_a(t)}{dt} = x_{a-1}(t)(x_{a+1}(t) - x_{a-2}(t)) - x_a(t) + \nu \quad (12)$$

and $a = 1, 2, \dots, D$; $x_{-1}(t) = x_{D-1}(t)$; $x_0(t) = x_D(t)$; $x_{D+1}(t) = x_1(t)$. ν is a fixed parameter which we take to be 10.0 where the solutions to the dynamical equations are chaotic [16]. The equations for the states $x_a(t)$; $a = 1, 2, \dots, D$ are meant to describe ‘weather stations’ on a periodic spatial lattice. This model is widely used in atmospheric science as a testbed for the exploration of innovative data assimilation ideas.

Our example selects $D = 11$, and displays the action level plot for $L = 2, 4, 5$, and 6 observations at each measurement time within the window $[t_0, t_F]$. We perform a ‘twin experiment’ wherein we generate D time series $\{x_a(t); a = 1, 2, \dots, D\}$ for Eq. (12) using a standard adaptive fourth order Runge-Kutta algorithm with a time step $\Delta t = 0.025$ and an initial condition $\mathbf{x}(t_0)$ drawn from a uniform distribution over the range of the variables $\mathbf{x}(t)$, namely $[-10, +10]$. To these solutions of Eq. (12) we add Gaussian noise with mean zero and variance $\sigma^2 = 0.2$ to each time series $x_a(t)$. These noisy

versions of our model time series constitute our ‘data’ $\{y_a(t)\}$. L of these D time series are presented to the model at times τ_n ; $t_0 \leq \tau_n \leq t_F$.

The measurement window is from $t_0 = 0$ to $t_F = 4.125$. $L = 2, 4, 5, 6$ ‘measurements’ are made at each time step; these are the $\mathbf{y}(\tau_n)$. The measurement error matrix \mathbf{R}_m is taken to have diagonal elements at each measurement time τ_n and is zero at other times. Its magnitude is taken as $R_m = 1/\sigma^2 = 5$.

The model error matrix $R_f(a)$ is also taken as diagonal, with elements along the diagonal $R_f = R_{f0}2^\beta$, in performing the VA procedure, and we take $\beta = 0, 1, 2, \dots$. R_{f0} was chosen 0.01.

The minimizations of nonlinear objective functions in the example using the Lorenz96 model was performed using the public domain software IPOPT [30] with a front end script written in Python.

In Fig. (1) we display action level plots for $L = 2, 3, 4$, and 6 observations at each measurement time. As we can see in the **TopLeftPanel**, where $L = 2$, there are numerous local minima in the action $A_0(\mathbf{X})$ for all values of $R_f \geq R_{f0}$, and these remain to large R_f . None of these minima is very far separated from the paths with the smallest minimum, so that the evaluation of the expected value integrals Eq. (11) would require contributions from many maxima of the conditional probability distribution.

When $L = 4$, **TopRightPanel** we begin to see an isolated action level whose contribution to the expected value integral is overwhelmingly larger than the contribution from path giving rise to the next largest action level. The value $L = 4$ is consistent with the observation in [16] that around $0.4D$ the instabilities in the Lorenz96 state space appear to be controlled by the data assimilation process.

At $L = 5$ or 6, **BottomPanels**, we see that the dominance of the lowest action level is even clearer. The horizontal olive colored line is the expected value of the measurement error term in the action. This is a sign of the consistency of the data assimilation calculations.

In Fig. (2) we explore another aspect of the action level plots. We still use $D = 11$, and we hold $L = 6$ fixed. The number of observations within the window $[t_0, t_F]$ has been reduced from 165 to 28 and we move the model forward between observations 0, 2, 5 or 11 times. This is to provide an analogy to how many layers are present in an equivalent machine learning example. Our example here differs by having many entry points in the measurement window while the machine learning example has only one. We display in the **LeftPanel** the action level plots for the selected number

of model evaluation steps. As one can see for 0 and 2 intermediate steps we have many persisting minima of the action. At 5 and 11 intermediate steps, there is only a single minimum that is found, and for large R_f it comes to the same action level as with 2 intermediate steps. All are consistent with the expected value of the measurement error term. This calculation, performed in a machine learning context provides information on how many hidden layers are required to achieve a desired accuracy.

In the **RightPanel** we display the accuracy of the estimation of the single parameter in the Lorenz96 model. It has been set at $\nu = 10.0$ in producing the data, and that value is clearly selected for 5 or 11 intermediate model evaluations, while it is not so clearly selected for 2 intermediate steps and with zero intermediate steps there is a consistent few percent error.

We see, in this collection of calculations, as noted earlier [32, 33], the ability to identify the dominant minimum of the action depends on the number of measurements presented during the statistical data assimilation procedure embodying transfer of information from the data to the model. In data assimilation this is associated with the number of positive conditional Lyapunov exponents [16] of the model. In the machine learning instantiation it may play the same role when the number of data presented at the output layer is not sufficient to determine the parameters and hidden states in each layer.

We also see the analogue of deepening the network produces higher accuracy estimates of conditional expected values.

5.2 A Multi-Layer Perceptron; Feedforward Network

We constructed a feedforward network with l_F layers: One input layer at l_0 and one output layer at l_F . This network has $l_F - 2$ hidden layers. We analyzed $l_F = 20, 30, 50$ and 100 . There are $N = 10$ ‘neurons’ in each layer. The activity in neuron j in layer l $x_j(l)$ is related to $x_j(l-1)$ in the previous layer as

$$x_j(l) = g(W(l)\mathbf{x}(l-1)); \quad g(z) = 0.5[1 + \tanh(\frac{z}{2})]. \quad (13)$$

We also investigated the “ReLU”-like function $g(z) = \log[1 + e^z]$, but we do not report on those results. The activations at the input layer are drawn from a Gaussian $N(0, 1)$. The weights are selected from a uniform distribution $U[-0.1, 0.1]$. Gaussian measurement noise was added to the ‘data’ generated by the model; this has zero mean and variance 0.0025.

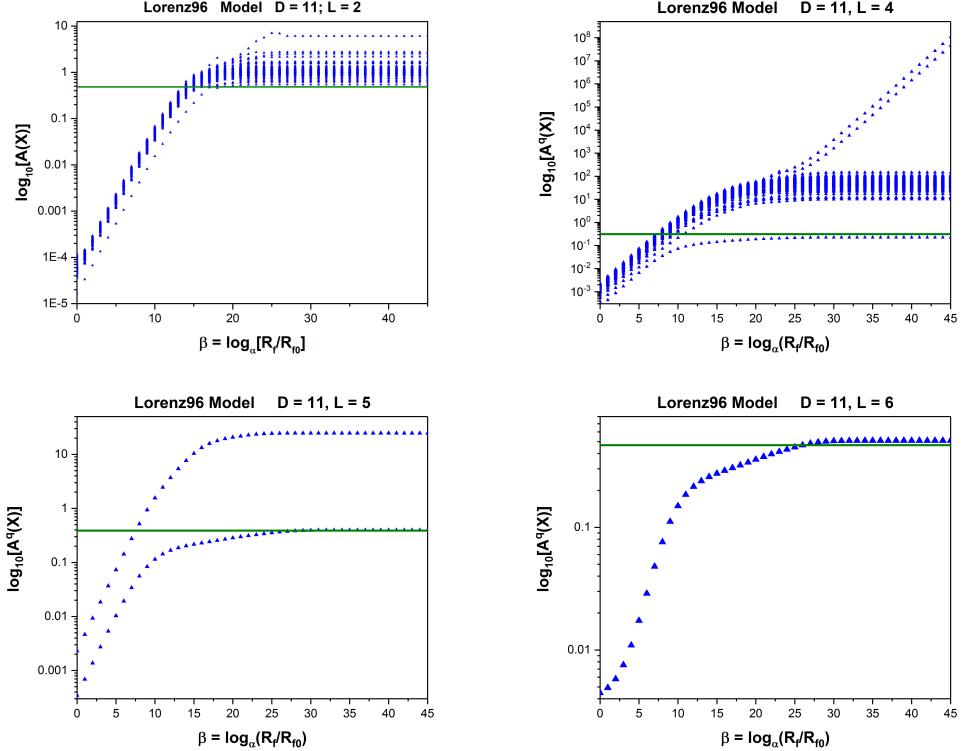


Figure 1: Action level plots for the Lorenz96 model Eq. (12) with $D = 11$ and $\nu = 10.0$. The variational annealing procedure is performed with 100 different initial conditions for the minimization of the action at each value of $R_f = R_{f0}\alpha^\beta$; $\beta = 0, 1, \dots$; $\alpha = 2$. **Top Left Panel:** $L = 2$ measurements at each measurement time. At $L = 2$, there are many minima, but none so much smaller than the others that it dominates the expected value integral Eq. (11). **Top Right Panel:** $L = 4$ measurements at each measurement time. At $L = 4$, the action in path space \mathbf{X} has numerous local minima. The lowest minimum has an action value much smaller than the action values from the other minima, and this dominates the expected value integral Eq. (11). **Bottom Left Panel:** $L = 5$ measurements at each measurement time. At $L = 5$, the number of minima found is only two, and again the lowest minimum dominates the expected value integral. **Bottom Right Panel** $L = 6$ measurements at each measurement time. At $L = 6$, there is only one minimum of the action. The solid green line is the expected value of the measurement error term. This is distributed as χ^2 . As the action becomes independent of R_f , its expected value should equal this value.

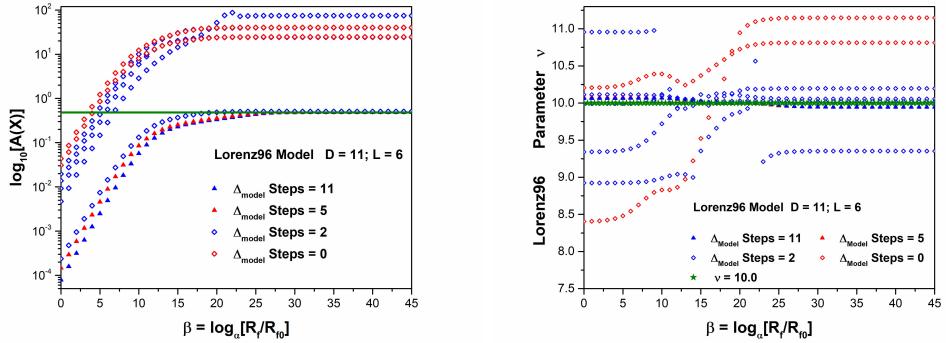


Figure 2: Parameter estimation and action level results for the Lorenz96 model, $D = 11$, $L = 6$. The parameter value $\nu = 10.0$ was used in the twin experiments on the model. Observations were made every $\Delta t_{obs} = 0.15$, and $L = 6$ measurements were made at each observation time. **Left Panel: Action Level Estimates** These are the action levels when $L = 6$ observations are made at each measurement time and with the choice of 0, 2, 5, and 11 model evaluation steps between measurement times. The horizontal olive green line indicates the expected action level for large R_f . **Right Panel Parameter estimates** Between the observations the model was evaluated 0, 2, 5 and 11 times leading to $\Delta t_{model} = 1.0, 0.33, 0.16$, and $0.08 * \Delta t_{obs}$. The parameter estimates are quite accurate for 5 and for 11 model time steps between observations. They are more distributed for 0 or 2 model step between observations. One can associate the idea of increasing the number of model steps between observations as equivalent to ‘deepening’ the hidden (unobserved) layers. The horizontal olive green line indicates the parameter value, 10.0, used in generating the data.

Our data are for the twin experiment, where we know all weights, all inputs and generate all values of $\mathbf{x}(l)$ in each layer $[l_0, l_F]$. These are recorded, and Gaussian noise with mean zero and variance 0.0025 is added to the data at layers $l_0 = 1$ and l_F starting from known values $\mathbf{x}(l_0)$. These data are $y_i(1)$ for the input layer and $y_i(20)$ for the output layer.

M input/output pairs are presented to the model with $L = 1, 5$, and 10 inputs $y_i^{(k)}(l_0)$ at layer l_0 and $L = 1, 5, 10$ data outputs $y_i^{(k)}(l_F)$ at layer l_F . $k = 1, 2, \dots, M$, and we investigated $M = 1, 10, 100$.

We minimize the action over all the weights and the states $x_a^{(k)}(l)$ at all layers of the model:

$$A_M(\mathbf{X}) = \frac{1}{M} \sum_{k=1}^M \left\{ \frac{R_m}{2L} \sum_{r=1}^L \left[(x_r^{(k)}(l_0) - y_r^{(k)}(l_0))^2 + (x_i^{(k)}(l_F) - y_i^{(k)}(l_F))^2 \right] \right. \\ \left. + \frac{R_f}{N(l_F - 1)} \sum_{l=1}^{l_F-1} \sum_{a=1}^N \left[x_a^{(k)}(l+1) - g(W(l)\mathbf{x}^{(k)}(l)) \right]^2 \right\}, \quad (14)$$

where we have $N = 10$ neurons in each layer and $L \leq N$ data at the input l_0 and at the output layers l_F .

We use the variational annealing procedure described above to identify the action levels for various paths through the network. The initial value of R_{f0}/R_m is taken to be 10^{-8} and this is incremented via $R_f/R_m = R_{f0}\alpha^\beta$ with $\alpha = 1.1$ and $\beta = 0, 1, \dots$.

In the numerical optimizations for the machine learning example we used L-BFGS-B [5, 34].

There are at least two ways to present more information to the model in this setting:

- increase the number of training pairs available to the network at l_0 and l_F ; this is our number M . M can be chosen as large as the user wishes.
- increase the number of components of the input/output pair vectors; this is our number L . $L \leq N$, the number of neurons in l_0 or l_F .

The resolution of the model in its ability to capture variations in activity from layer to layer is improved by increasing the number of layers l_F .

In Fig. (3) we explore the action levels as a function of R_f/R_m as we increase the number of layers l_F in the model: $l_F = 20, 30, 50, 100$. We hold fixed the number of neurons $N = 10$, the number of training pairs $M = 100$ and the number of inputs and outputs $L = 10$ at l_0 and l_F .

In each case there are many local minima, but only when $l_f = 50$ does the lowest action minimum significantly split from the other action minima and qualify to dominate the expected value integral Eq. (11). When l_F is increased from 50 to 100, the lowest action minimum comes closer to the second lowest minimum. This is seen as a result of the much larger number of weights to be estimated at the latter value of l_F while we are holding fixed through the values of L and M the information available to make those estimations.

In Fig. (4) we hold M fixed at 100, and l_F fixed at 50 while we look at $L = 1, 5, 10$ values of the dimension of input/output pairs.

In Fig. (5) we hold fixed the number of layers l_F , the number of neurons in each layer $N = 10$ and the dimension of the input/output vectors $y_r^{(k)}(l_0)$ and $y_r^{(k)}(l_F)$; $r = 1, 2, \dots, L = 10$. We show the effect of increasing the number of input/output pairs from $M = 1$ to $M = 10$ to $M = 100$. The emergence of a lowest action minimum as M increases is displayed. This can serve as a candidate for approximating Eq. (11).

In Fig. (6) we display the error in prediction after the mode, with $l_F = 50$ layers, has been set by the estimation of the weights. This error is constructed by selecting M_P new input output pairs. Using each of the input elements for $L \leq N$ components, we use the model with our estimated weights to evaluate $x_r^{(k)}(l_F)$ and compare that with $y_r^{(k)}(l_F)$ from each of the M_P pairs. The square error averaged over L presented components and over M_P pairs

$$\frac{1}{LM_P} \sum_{k=1}^{M_P} \sum_{r=1}^L (x_r^{(k)}(l_F) - y_r^{(k)}(l_F))^2 \quad (15)$$

is displayed.

We see that increasing the information presented via increasing L or M leads to decreased average prediction errors when choosing the path corresponding to the lowest action level, **Top Layers**, or choosing the path associated with the second lowest action level **Bottom Panel**. The differences in quality of prediction (or generalization) in these examples among the cases analyzed is not large, and this has been noted [12].

5.3 Recurrent Networks

In this network architecture one allows both interactions among neurons from one layer to another layer as well as interactions among neurons within

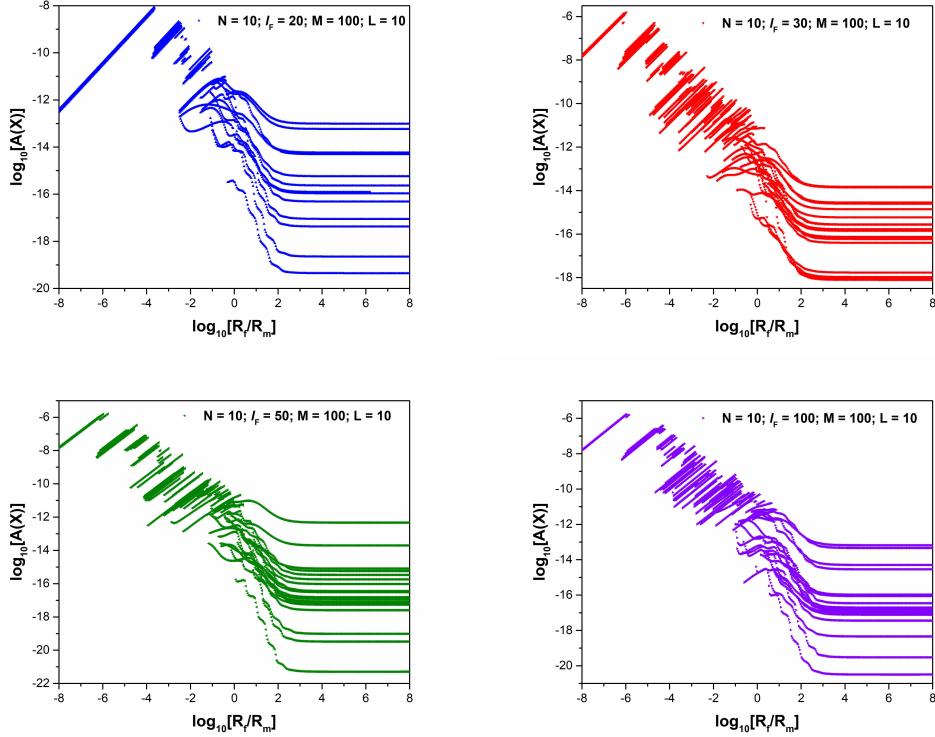


Figure 3: Holding the number of neurons in each layer fixed at 10, the number of input/output pairs fixed at $M = 100$, and the number of inputs and outputs at l_0 and l_F fixed at $L = 10$, we vary the number of layers (the deepening of the network) and examine the action level plots arising from the variational annealing procedure. **Upper Left Panel** $l_F = 20$ **Top Right Panel** $l_F = 30$ **Bottom Left Panel** $l_F = 50$ and **Bottom Right Panel** $l_F = 100$.

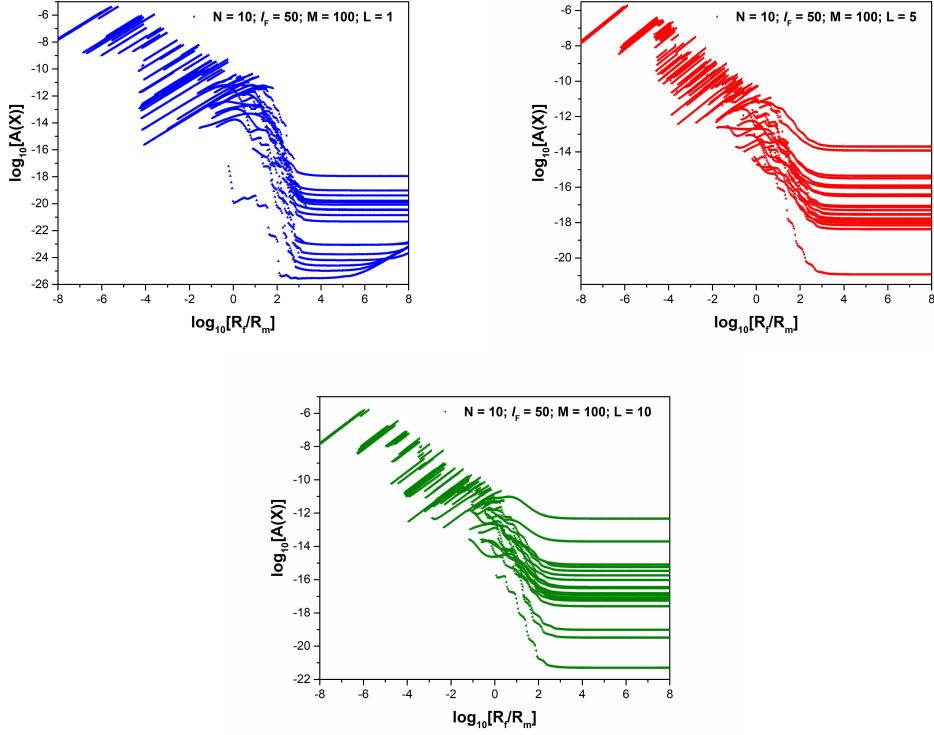


Figure 4: Holding the number of neurons fixed $N = 10$, the number of layers fixed $l_F = 50$, and the number of input/output training pairs fixed $M = 100$, we display the action levels as we vary the number of inputs at l_0 and the number of outputs at l_F . **Top Left Panel** $L = 1$. **Top Right Panel** $L = 5$. **Bottom Panel** $L = 10$.

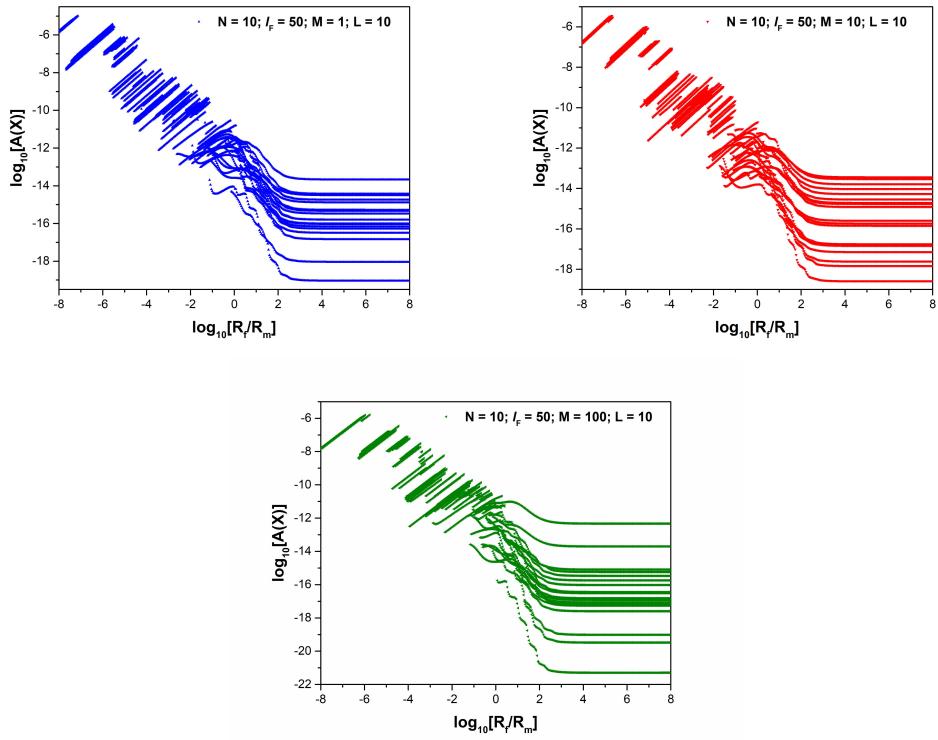


Figure 5: Holding the number of neurons fixed $N = 10$, the number of layers fixed $l_F = 50$, and the number of inputs $L = 10$ at l_0 and the number of outputs at l_F , we display the action levels as we vary the number of training pairs M . **Top Left Panel** $M = 1$. **Top Right Panel** $M = 10$. **Bottom Panel** $M = 100$.

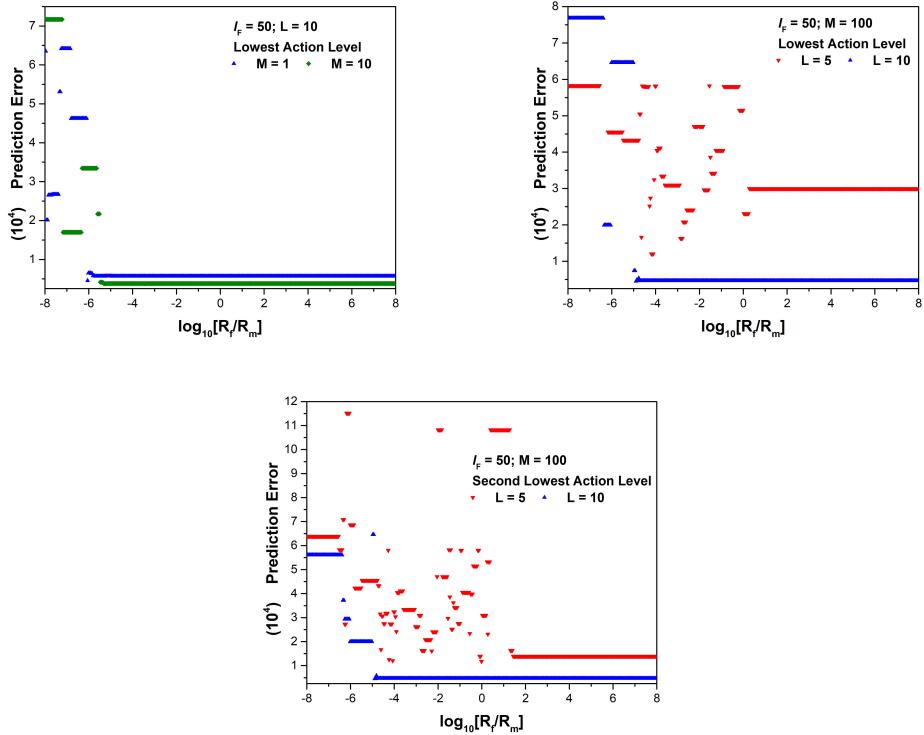


Figure 6: Prediction Errors for the AI/Machine Learning Network $\frac{1}{LM_P} \sum_{k=1}^{M_P} \sum_{r=1}^L (x_r^{(k)}(l_F) - y_r^{(k)}(l_F))^2$ averaged over M_P new input/output pairs. A noisy input $y_r^{(k)}(l_0)$ produces the output $x_r^{(k)}(l_F)$ using the model estimated using the M training sets. This is compared with the output $y_r^{(k)}(l_F)$ produced with the original model used to produce the ‘data’ for the twin experiment. In each case the number of neurons is $N = 10$ and $l_F = 50$. **Top Left Panel** Using the model associated with lowest Action Level: $L = 10$ and $M = 1$ and $M = 10$. **Top Right Panel** Using the model associated with lowest Action Level: $L = 5$ and $L = 10$; $M = 100$. **Bottom Panel** Using the model associated with the second lowest Action Level: $L = 5$ and 10 ; $M = 100$.

a single layer [14, 8]. The activity $x_j(l)$ of neuron j , $j = 1, 2, \dots, N$ in layer l $\{l_0, l_1, \dots, l_F\}$ is given by $x_j(l) = f[\sum_i w_{ji}(l)x_i(l-1)]$ in a feedforward, **layer goes to the next layer**, network.

We can add interactions with a layer in the same fashion, and to give some ‘dynamics’ to this within-layer activity we introduce a sequential label σ to the activity of neuron j in layer l : $x_j(l, \sigma)$. The mapping from layer to layer and within a layer can be summarized by

$$x_j(l, \sigma) = f[\sum_i w_{ji}(l)x_i(l-1, \sigma) + \sum_i W_{ji}(l)x_j(l, \sigma-1)], \quad (16)$$

Another version of this allows the nonlinear function to be different for layer-to-layer connections and within-layer connections, so

$$x_j(l, \sigma) = f[\sum_i w_{ji}(l)x_i(l-1, \sigma)] + g[\sum_i W_{ji}(l)x_j(l, \sigma-1)], \quad (17)$$

where $f(x)$ and $g(x)$ can be different nonlinear functions.

We can translate these expressions into the DA structure by recognizing that $x_j(l)$ is the ‘model variable’ in the layer-to-layer function while in the recurrent network, the state variables become $x_j(l, \sigma)$. It seems natural that as dimensions of connectivity are added—here going from solely feedforward to that plus within-layer connections—that additional independent variables would be aspects of the ‘neuron’ state variables’ representation.

In adding connections among the neurons within a layer we have another independent variable, we called it σ , and the ‘point’ neurons depending on layer alone become **fields** $x_j(l, \sigma)$. In the machine learning/AI networks we have no restrictions on the number of independent variables. This may lead to the investigation of ‘neural’ fields $\phi_j(\mathbf{v})$ where \mathbf{v} is a collection of independent variables indicating which layers are involved in the progression of the field from an input to an output layer.

However many independent variables and however many ‘neurons’ we utilize in the architecture of our model network, the overall goal of identifying the conditional probability distribution $P(\mathbf{X}|\mathbf{Y})$ and estimating the moments or expected values of interest still comes down to one form or another in the approximation of integrals such as Eq. (6).

5.4 Making Time Continuous; Continuous Layers: Deepest Learning

There is much to learn about the data assimilation or machine learning problem as the number of layers or equivalently the number of time points within an epoch becomes very large. The limit of the action where the number of layers in an epoch becomes a continuous variable is, in data assimilation notation [15],

$$A_0(\mathbf{x}(t), \dot{\mathbf{x}}(t)) = \int_{t_0}^{t_F} dt L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)$$

$$L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) = \sum_{r=1}^L \frac{R_m(l, t)}{2} \left(x_r(t) - y_r(t) \right)^2 + \sum_{a=1}^D \frac{R_f(a)}{2} \left(\dot{x}_a(t) - F_a(\mathbf{x}(t)) \right)^2.$$

In this formulation the quantity $R_m(l, t)$ is non-zero only near the times $t \approx \tau_k$. It can be taken as proportional to $\delta(t - \tau_k)$.

Within the machine learning context we call this ‘deepest learning’ as the number of layers goes to infinity in a useful manner.

The minimization of the action now requires that the paths $\mathbf{x}(t)$ in $\{\mathbf{x}(t), \dot{\mathbf{x}}(t)\}$ space satisfy the Euler-Lagrange equation $\frac{d}{dt} \left[\frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial \dot{x}_a(t)} \right] = \frac{\partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t)}{\partial x_a(t)}$, along with the boundary conditions $\delta x_a(t_0) p_a(t_0) = 0$; $\delta x_a(t_F) p_a(t_F) = 0$ where $p_a(t) = \partial L(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) / \partial \dot{x}_a(t)$ is the canonical momentum.

For the standard model, the Euler-Lagrange equations take the form

$$R_f \left[\frac{d}{dt} \delta_{ab} + D F_{ab}(\mathbf{x}(t)) \right] \left[\frac{dx_b(t)}{dt} - F_b(\mathbf{x}(t)) \right] = R_m(l, t) \delta_{ar} (x_r(t) - y_r(t)), \quad (18)$$

where we have $\mathbf{DF}(\mathbf{x}) = \partial \mathbf{F}(\mathbf{x}) / \partial \mathbf{x}$.

The E-L equations are the necessary condition, along with the accompanying boundary conditions, that show how errors represented on the right hand side of the E-L equation drive the model variables at all layers to produce $\mathbf{x}(l) \rightarrow \mathbf{y}(l)$ where data is available.

In the integral for $\langle G(\mathbf{X}) \rangle$, the coordinates $\mathbf{x}(t_0)$ and $\mathbf{x}(t_F)$ are not restricted, so we have the ‘natural’ boundary conditions [11, 17, 22] $p_a(t_0) = 0$ and $p_a(t_F) = 0$.

This shows quite clearly that the minimization problem requires a solution of a two point boundary value problem in $\{\mathbf{x}(t), \mathbf{v}(t) = \dot{\mathbf{x}}(t)\}$ space. One way to address two point boundary value problems is to start at one end, t_0 with a value of $\mathbf{x}(t_0)$ and proceed from t_F with a value of $\mathbf{x}(t_F)$ and integrate both

ways requiring a match [27]. Furthermore, the residual of the measurement error term on the right hand side of Eq. (18) nudges the solution in $\mathbf{x}(t)$ to the desired output.

If one were to specify $\mathbf{x}(t_0)$, but not $\mathbf{x}(t_F)$, then the boundary conditions for the Euler-Lagrange equation are the given $\mathbf{x}(t_0)$ ($\delta\mathbf{x}(t_0) = 0$) and require the canonical momentum $p_a(t_F) = 0$. Examining the Hamiltonian dynamics for this problem then suggest integrating the $\mathbf{x}(t)$ equation forward from t_0 and the canonical momentum equation backward from t_F . This is back propagation.

5.4.1 Hamiltonian Dynamics Realization

If one moves from the Lagrangian realization of the variational problem to a Hamiltonian version by trading in the phase space from $\{\mathbf{x}(t), \mathbf{v}(t)\}$ to canonical coordinates $\{\mathbf{x}(t), \mathbf{p}(t)\}$, then the Hamiltonian $H(\mathbf{x}, \mathbf{p})$ for the standard model reads

$$H(\mathbf{x}, \mathbf{p}, t) = \sum_{a=1}^D \left\{ \frac{p_a(t)p_a(t)}{2R_f(a)} + p_a(t)F_a(\mathbf{x}(t)) \right\} - \sum_{r=1}^L \frac{R_m(r, t)}{2}(x_r(t) - y_r(t))^2. \quad (19)$$

In these coordinates the equations of motion are then given by Hamilton's equations

$$\begin{aligned} \frac{dp_a(t)}{dt} &= -p_b(t) \frac{\partial F_b(\mathbf{x}(t))}{\partial x_a(t)} + \delta_{ar} R_m(r, t)(x_r(t) - y_r(t)) \\ \frac{dx_a(t)}{dt} &= F_a(\mathbf{x}(t)) + \frac{p_a(t)}{R_f(a)}. \end{aligned} \quad (20)$$

Returning from this to discrete time (or layers) we see that if the variational principle is carried out in $\{\mathbf{x}, \mathbf{p}\}$ space, the boundary conditions $p_a(t_0) = p_a(t_F) = 0$ are quite easy to impose while the other variables, all the $x_a(t_k)$ and the $p_a(t_k)$; $k \neq 0, F$, are varied. Going forward in \mathbf{x} and backward in \mathbf{p} is neither required nor suggested by this formulation. It is worth noting that in either $\{\mathbf{x}, \mathbf{v}\}$ space or $\{\mathbf{x}, \mathbf{p}\}$ space, the continuous time (layer) formulation has a symplectic symmetry [11, 15]. This not automatically maintained when the discrete time (layer) problem is reinstated [24, 31]; however, many choices of integration procedure in which time/layer becomes discrete and the symplectic symmetry is maintained are known [24, 31, 13].

In a detailed analysis [15, 33] of the variational problem in Lagrangian and Hamiltonian formulations, it appears that the direct Lagrangian version

in which the state variables $\mathbf{x}(t_n)$ or $\mathbf{x}(l_n)$ are varied, the symplectic structure can be maintained and the boundary conditions on the canonical momentum respected [24, 31].

In practice, this means that the direct variational methods suggested for the machine learning problems taking into account model error ($R_f \neq \infty$) may skirt issues associated with back propagation. This issue may be seen a bit more directly by comparing how one moves in $\{\mathbf{x}(t), \dot{\mathbf{x}}(t)\}$ space organized by Eq. (18) with the motion in $\{\mathbf{x}(t), \mathbf{p}(t)\}$ space guided by Eq. (20). These are equivalent motions of the model in time/layer, connected by a Legendre transformation from $\{\mathbf{x}(t), \dot{\mathbf{x}}(t)\} \rightarrow \{\mathbf{x}(t), \mathbf{p}(t)\}$.

In the Hamiltonian form, where $R_f \rightarrow \infty$ is the limit where one usually works, moving in regions where $\mathbf{DF}(\mathbf{x})$ may have saddle points may ‘slow down’ the progression in the canonical momentum $\mathbf{p}(t)$. This may occur at a maximum, at a minimum, or at a saddle point of $\mathbf{DF}(\mathbf{x})$. At any of these the observation in [21]: “The analysis seems to show that saddle points with only a few downward curving directions are present in very large numbers, but almost all of them have very similar values of the objective function. Hence, it does not much matter which of these saddle points the algorithm gets stuck at.” may apply. In the Lagrangian formulation Eq. (18) the manner in which $\mathbf{DF}(\mathbf{x})$ enters the motion is quite different and may well avoid this confounding property. We have pointed out that Eq. (20) is backprop. The use of the Lagrangian variational principle [24, 31] solves the same problem, so may have an unexpected virtue.

6 Summary and Discussion

This paper has been directed to drawing a direct analogy between the formulation of a much utilized class of machine learning problems and a set of equivalent problems in data assimilation as encountered in many physical, biological and geoscience problems as well as in many engineering analyses where data driven model development and testing is a goal. The fundamental equivalence of the two inquiries is the core of this paper.

The analogy allows us to identify methods developed in data assimilation as potentially quite useful in machine learning contexts. In particular the possibility of using variational annealing to produce the global minimum of the action (cost function) of the standard model of data assimilation with both observation error and model error appears potentially of value.

The idea of making time continuous for purposes of exploring properties of data assimilation suggests a similar tactic in machine learning. The machine learning step of making layers continuous we have called “deepest learning” as deep learning appears to result from increasing the number of layers. In the continuous layer (time) formulation, we see clearly that the problem to be solved is a two point boundary value problem. This may lead to the construction and solution of tractable models that may helpfully illuminate how deep learning networks operate successfully and expand the possibilities of utilizing them employing additional methods for numerical calculations and for interpretation.

In the formulation of the statistical data assimilation problem at the general level expressed in Eq. (5) we see that the measurement error term which is where information from data is passed to the model, it is explicitly information through the conditional mutual information that is being passed from observations to the model. This suggests that the idea that deep learning works because of major increases in computing power as well as in having large data sets, however, the attribute of the data sets is not so much as they are large but that they possess information, in a precise manner, that can be utilized to learn about models. The conjunction of information transfer and state and parameter estimation is embodied in the work of Rissanen [28, 29] where he identifies the cost of estimating a parameter or a state variable at some time. The arguments we have presented suggest evaluating how much information in a data set is available to inform a model is of greater utility than just the size of the data set itself.

One point not made explicit in the main text, but worth noting, is that once we have formulated the data assimilation or machine learning problems as accurately performing high dimensional integrals such as Eq. (6), the Laplace approximation method, namely the usual variational principle, permits the investigation of corrections through further terms in the expansion of the action about the path leading to the global minimum. In [33] it is shown that corrections to this first approximation are small as R_f becomes large when analyzing the standard model. This need not be the case for other choices of noise distributions in the measurement error or model error terms in the action.

Another item of interest is the argument noted in [21] that as the dimension of a model increases, one may find fewer and fewer local minima confounding the search in path space for a global minimum, and in that situation many more unstable saddle points in path space will arise [7, 6].

In the case of a chaotic system such as the Lorenz96 model, the evidence is that however large the dimension of the model itself and the paths over which one may search, there are multiple local minima until the number of measurements at any observation time is large enough and the information transferred to the model is sufficient. The role of the number of model evaluations between observations, suggested in some of the arguments here, also play a significant part in establishing whether the action surface has many local minima.

The view of a deep network as moving from a few hidden layers to many may also be illuminated by our arguments. One idea is that by increasing the number of hidden layers one is increasing the resolution in the analog of ‘time’ in data assimilation. When one does that in data assimilation, we see it as probing the variation of the underlying model as it evolves from an initial condition through ‘layer’ = ‘time.’ Missing the higher frequency variations in time by employing a coarse grid in discrete time should have its counterpart role in the feedforward networks discussed here.

It is recognized that the ‘neuron’ models widely utilized in machine learning applications have little in common with properties of biological neurons, the construction and implementation of large networks that have successful function within machine learning may prove a useful guide for the construction and implementation of functional natural neural networks.

Finally, it is important to comment that while the analogy drawn and utilized here may improve the testing and validation of models supported by observational data, it does not assist in the selection of the models and their formulation. That is still a task to be addressed by the user.

Acknowledgments

We express our appreciation to our colleagues Dan Breen and Jeff Elman for important discussions. Also the CENI team made finding the analogs in this paper possible. Many thanks to Tim Gentner, Gert Cauwenberghs, and especially Gabe Silva improved the manuscript. Jon Shlens provided a detailed view from the active world of machine learning. Partial support from the MURI Program (N00014-13-1-0205) sponsored by the Office of Naval Research is acknowledged as is support for A. Shirman from the ARCS Foundation.

References

- [1] *Perspectives on Research in Artificial Intelligence and Artificial General Intelligence Relevant to DoD*, 2017. https://www.google.com/?gws_rd=ssl#q=perspectives+on+research+in+artificial+intelligence+and+artificial+general+intelligence.
- [2] Henry D. I. Abarbanel. *Predicting the Future: Completing Models of Observed Complex Systems*. Springer, 2013.
- [3] E. L. Allgower and K. Georg. *Numerical Continuation Methods: An Introduction*. Springer-Verlag, 1990.
- [4] A. F. Bennett. *Inverse Methods in Physical Oceanography*. Cambridge University Press, 1992.
- [5] R. H. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16:1190–1208, 1995.
- [6] Anna Choromanska, Mikael Henaff, Michael Mathieu, Ge'rard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *18th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2015.
- [7] Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 2933–2941, Cambridge, MA, USA, 2014. MIT Press.
- [8] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
- [9] Geir Evensen. *Data Assimilation: The Ensemble Kalman Filter*. Springer, 2009.
- [10] Robert M. Fano. *Transmission of Information; A Statistical Theory of Communication*. MIT Press, 1961.

- [11] I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Dover Publications, Inc., 1963.
- [12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA; London, UK, 2016. <http://www.deeplearningbook.org>.
- [13] Ernst Hairer, Gerhard Wanner, and Christian Lubich. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*. Springer Series in Computational Mathematics; Volume 31, 2nd edition, 2006.
- [14] M. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Conference of the Cognitive Science Society*, pages 531–546. Cognitive Science Society, 1986.
- [15] N. Kadakia, D. Rey, J. Ye, and H. D. I. Abarbanel. Symplectic structure of statistical variational data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 143(703):756–771, 2017.
- [16] Mark Kostuk. Synchronization and statistical methods for the data assimilation of hvc neuron models. *PhD Dissertation in Physics, University of California, San Diego*, 2012.
- [17] Mark Kot. *A First Course In the Calculus of Variations*. American Mathematical Society; Providence, Rhode Island., 2014.
- [18] Rubin H. Landau, Manuel Jose' Paez, and Cristian C. Bordeianu. *A Survey of Computational Physics: Introductory Computational Science*. Princeton University Press, 2010.
- [19] Pierre Simon Laplace. Memoir on the probability of causes of events. *Mémoires de Mathématique et de Physique, Tome Sixième*, pages 621–656, 1774.
- [20] Pierre Simon Laplace. Memoir on the probability of the causes of events. *Statistical Science*, 1(3):364–378, 1986. Translation to English by S. M. Stigler.
- [21] Yann LeCunn, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

- [22] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory*. Princeton University Press, 2012.
- [23] Edward N Lorenz. Predictability: A problem partly solved. In Tim Palmer and Renate Hagedorn, editors, *Predictability of weather and climate*. Cambridge, 2006.
- [24] J. E. Marsden and M. West. Discrete mechanics and variational integrators. *Acta Numerica*, pages 357–514, 2001.
- [25] K. G. Murty and S. N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987.
- [26] Alexander G. Parlos, Kil T. Chong, and Amir F. Atiya. Application of the recurrent multilayer perceptron in modeling complex process dynamics. *IEEE Transactions on Neural Networks*, 5:255–266, 1994.
- [27] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing, Third Edition*. Cambridge University Press, 2007.
- [28] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry Theory*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1989.
- [29] Jorma Rissanen. *Information and Complexity in Statistical Modeling*. Springer, 2007.
- [30] A. R. Wachter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006.
- [31] J. M. Wendlandt and J. E. Marsden. Mechanical integrators derived from a discrete variational principle. *Physica D: Nonlinear Phenomena*, 106:223–246, 1997.
- [32] J. Ye, N. Kadakia, P. J. Rozdeba, H. D. I. Abarbanel, and J. C. Quinn. Improved variational methods in statistical data assimilation. *Nonlinear Processes in Geophysics*, 22(2):205–213, 2015.

- [33] J. Ye, Daniel Rey, Nirag Kadakia, Michael Eldridge, Uri Morone, Paul Rozdeba, Henry D. I. Abarbanel, and John C. Quinn. A systematic variational method for statistical nonlinear state and parameter estimation. *Physical Review E*, 2015.
- [34] C. Zhu, R. H. Byrd, and J. Nocedal. L-bfgs-b: Algorithm 778: L-bfgs-b, fortran routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23:550–560, 1997.