nature reviews physics

https://doi.org/10.1038/s42254-024-00776-3

Supplementary information

Machine learning for the physics of climate

In the format provided by
the authors and unedited

Supplementary information for Machine Learning for the Physics of Climate

Annalisa Bracco[1,*], Julien Brajard[2], Henk A. Dijkstra[3], Pedram Hassanzadeh[4], Christian Lessig[5], and Claire Monteleoni[6,7]

[1]School of Earth and Atmospheric Sciences, Georgia Institute of Technology, Atlanta, GA, USA
[2]Nansen Environmental and Remote Sensing Center (NERSC), Bergen, Norway
[3]Institute for Marine and Atmospheric research, Utrecht University, Utrecht, the Netherlands
[4]Department of Geophysical Sciences and Commitee on Computational and Applied Mathematics, University of Chicago, IL, USA
[5]European Centre for Medium Range Weather Forecasts (ECMWF), Reading, UK
[6]INRIA Paris, France
[7]Computer Science Department, University of Colorado Boulder, Boulder, CO, USA
[*]e-mail:abracco@gatech.edu

Methods: Machine Learning Techniques

Machine Learning is a broad field encompassing many methods, several of which have been applied to climate science. The list includes deep learning, probabilistic graphical models, causal analysis, online learning, clustering, and ensemble methods such as random forests. Although our Review references multiple types of machine learning, given the breadth of the field, we will limit this section mostly to neural networks, the basis of the modern field of deep learning.

Neural networks can be understood as parameterized nonlinear mappings $\mathscr{N}_\theta : V \to W$ between finite dimensional vector spaces V, W. By choosing a basis, the network can thus be seen as a map $\mathscr{N}_\theta : \mathbb{R}^n \to \mathbb{R}^m$ (with the usual identifications for complex or other vector spaces). The parameters are denoted collectively as θ, with $\theta \in \mathbb{R}^K$, and they are determined to (approximately) optimally fit a relationship in observed data, a process known as neural network training.

Neural network architectures Neural networks are usually composed of a set of layers, that is $\mathscr{N}_\theta = L_N \circ \cdots \circ L_1$, which simplifies their practical implementation and allows for efficient computations highly optimized for a small set of standard layers and layer components. The classical "perceptron" is a feed-forward layer of the form $L_i(x) = \sigma(W_i x + b_i)$ where $x \in \mathbb{R}^{n_i}$, $W \in \mathbb{R}^{n_i \times m_i}$ is the weight matrix, $b_i \in \mathbb{R}^{m_i}$ the bias vector, and $\sigma(\cdot)$ is a nonlinear function applied element-wise to the affine transformation $W_i x + b_i$. Common examples for $\sigma(\cdot)$ are the sigmoid function, given by $e^x/(1+e^x)$, for scalar x, or the ReLU function, which is zero for negative arguments and the linear function x otherwise. It is often informative to consider W_i as the adjacency matrix for a graph that "routes" the data through the network.

The composition of multiple layers of the above type is known as a multi-layer perceptron or a feed-forward neural network. The trainable network parameters θ are the entries of the weight matrices W_i and bias vectors b_i. Another common neural network layer L_i is a convolutional one, which is similar to a feed-forward layer but where W_i is a circulant matrix that implements a (discrete) convolution. Convolutional layers substantially reduce the number of parameters compared to multi-layer perceptrons and they are useful for data on a regular grid with an approximate translation invariance, such as images.

Another important layer type is self-attention. It takes as input a set or sequence $\{x_i\}_{i=1}^r$ of so called tokens $x_i \in \mathbb{R}^{n_i}$ and outputs an updated set or sequence $\{\tilde{x}_i\}_{i=1}^r$ given by

$$\tilde{x}_i = \sum_{j=1}^{r} \rho\left(x_i^T W_q^T W_k x_j\right)(W_v x_j) \tag{1}$$

where W_q, W_k, W_v are projection matrices and ρ is the softmax function, the generalization of the logistic sigmoid function (used in probabilistic, binary classification) to multiple classes. This can be thought of as smoothed version of the argmax that also ensures that the output is normalized and can be interpreted as a probability distribution. In essence, attention layers update each x_i based on the dot product similarity to all other tokens and filtered by the softmax nonlinear function. A full attention layer usually consists of multiple parallel (and hence independent) so-called attention heads performing the computations in Eq. 1 with independent projections W_q, W_k, W_v.

Transformers, which are one of the most popular and successful neural network architectures to-date, consist of alternating attention and multi-layer perceptron layers. Conceptually, these update tokens with the inter-token attention computations in Eq. 1 and then map all tokens independently to a new vector space where the attention computation can be repeated, revealing

different information. The layers are combined with so-called skip connections so that only a residual update is determined through the layers: $\bar{x}_j = x_j + L_i(x_j)$. Skip connections are typically used to improve the numerical stability of the training process. A variation of classical transformers is Fourier neural operators where the attention is computed in frequency space, exploiting the fact that a translation-invariant integral kernel can be expressed efficiently in the Fourier domain.

Neural network training Training of neural networks refers to the nonlinear optimization that is used to determine a set of optimal (or suitable) network parameters θ given a set of data $\mathscr{D}$. In the simplest case, $\mathscr{D}$ consists of pairs $\mathscr{D} = (x_i, y_i)_{i=1}^{D}$ and the y_i are the target quantities of interest that are to be estimated or predicted by the neural network. For example, the x_i can be images and the y_i class labels that determine the image content. Directly training x_i towards y_i is known as supervised learning. When the y_i are proxies to train the neural network but not themselves of interest then one speaks of self-supervised training. This is typically complemented with a second training phase towards a specific task or application. The purpose of self-supervised learning is to learn overall domain knowledge so that the model can represent the underlying structure and patterns of the data without relying on explicit labels. A classical example of self-supervised training is image inpainting, where parts of images are masked and the self-supervised training task of the network is to predict these parts.

The optimization that is required for training is typically solved with stochastic gradient descent where a Monte Carlo estimate of the gradient over a subset of the data is used in each gradient descent step. The Monte Carlo estimate is used because it is computationally cheaper and requires less computer memory than optimizing over the full training set in one batch. Empirically, it is also key to a successful optimization of neural networks with millions, billions or even trillions of trainable parameters because the stochasticity helps to avoid ineffective local minima. A wide range of energy or loss functions are used for the gradient descent. When the $y_i \in \mathbb{R}^n$, then the mean squared error is, arguably, the most common loss function. When y_i is from a discrete set, then cross-entropy is commonly used as loss.

A common problem with neural network training is overfitting, which is identified when the network performs well on the training set but not on new (unseen) data. To diagnose overfitting and enable an estimate of the performance of a trained neural network in applications, the data set is typically partitioned into training, validation, and test sets. Computing the skill of the neural network on the validation set is used to monitor training progress without affecting (and hence biasing) the network weights. In other words, the validation step helps with detecting overfitting or selecting hyperparameters, such as the number of layers and parameters in the neural network or the parameters of the stochastic gradient descent. The test set is employed only after training, to provide a final evaluation of the network performance in inference, that is when the trained network with fixed weights is put to its intended use (for example for prediction, estimation, or classification, etc.). Common strategies to address overfitting are to restrict the network size or use drop-out, which occurs when a random subset of the training weights is set to zero in each training optimization step. For sufficiently large training datasets, however, network size is no longer directly related to overfitting. Often very large (even over-parameterized) networks perform best[1], because parts of a large network can internally specialize for processing only specific inputs, for example a certain input variable at a specific height level in a weather model.

Generative models For classification problems, such as predicting the class of an image, network output is typically modeled not as a discrete value but as a probability distribution over all possible labels. This can be realized by predicting $y_i \in R^M$, where M is the number of classes, and then normalizing the values to obtain a probability distribution, such as with the softmax function. The network can then be interpreted as modeling the conditional probability distribution $p(y|x)$ of outputs y given the inputs x. The perspective of a neural network modeling a conditional or joint probability distribution is widely used in the machine learning literature and the corresponding models are referred to as generative models.

This principle can be extended to continuous probability functions p, with the network being trained to generate either moments of the probability distribution or samples from this distribution. For multi-dimensional data such as images or physical fields, generative adversarial networks (GANs) and diffusion models are common generative architectures with strong theoretical foundations that can be interpreted from a statistical mechanical point of view. Any regression problem trained with mean squared error can be interpreted as a generative model where only the mean of a Gaussian distribution is considered.

Incorporating physics into neural networks Neural networks extract their power from the training data through the training process, with only weak priors through the network architecture (for example when a convolutional network is used). This stems largely from their development being driven by natural language processing and computer vision where no simple analytic theory is available. It is possible to incorporate prior physical knowledge following strategies explored more generally for physical systems[2]. Known symmetries[3,4], autocorrelations[5], and conservation laws[6] have been incorporated into neural networks. Physics-informed neural networks (PINNs) replace the training set $\mathscr{D}$ with a known constraint that the neural network output has to satisfy, typically a known physical equation[7].

Reservoir Computing A special class of methods are recurrent neural networks. In contrast to uni-directional feedforward neural networks, these networks are bi-directional, meaning that they allow the output from some nodes to affect subsequent

input to the same nodes. Recurrent neural networks are difficult to train and several variant have been developed to circumvent that problem, such as Long-Short Term Memory networks and Reservoir Computing[8]. In the latter method, the input time series is mapped to a high-dimensional dynamical system (the reservoir) and only the output weights (connecting the reservoir nodes and the output) are adjusted during training. Reservoir Computing therefore has a close connection to dynamical systems theory and has been widely used in climate prediction studies[9,10].

Explainability and interpretability There has been much research in machine learning on how to interpret the learned models. Some models are by nature interpretable (for example, linear regression yields a weighting over its input variables), but for others, especially deep neural networks, methods to provide post hoc "explanations" for model decisions have been developed. The AI subfield of Explainable AI (XAI) is concerned with understanding the reason why a neural network output was generated. Post hocanalysis of a deep neural network via an XAI technique can help assess the model's confidence in the decision, detect cases of inappropriate usage, and even derive physical insights from the neural network models themselves. In climate physics, where understanding is often more important than quantitative performance, XAI is a particularly active field [11–13]. We refer to[14] for a discussion of the challenges in choosing the best XAI method in the domain of climate science and for an extensive list of references.

References

1. Kaplan, J. *et al.* Scaling laws for neural language models. *arXiv preprint arXiv:2401.05932* DOI: 10.48550/ARXIV.2001.08361 (2020).
2. de Bezenac, E., Pajot, A. & Gallinari, P. Deep learning for physical processes: Incorporating prior scientific knowledge. *J. Stat. Mech. Theory Exp.* **2019**, DOI: 10.1088/1742-5468/ab3195 (2019).
3. Hutchinson, M. *et al.* Lietransformer: Equivariant self-attention for lie groups. *arXiv preprint arXiv:2012.10885* DOI: 10.48550/arXiv.2012.10885 (2021).
4. Satorras, V. G., Hoogeboom, E. & Welling, M. E(n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844* DOI: 10.48550/arXiv.2102.09844 (2022).
5. Falasca, F., Bracco, A., Nenes, A. & Fountalis, I. Dimensionality reduction and network inference for climate data using -maps: Application to the cesm large ensemble sea surface temperature. *J. Adv. Model. Earth Syst.* **11**, 1479–1515, DOI: https://doi.org/10.1029/2019MS001654 (2019).
6. Beucler, T. *et al.* Enforcing analytic constraints in neural networks emulating physical systems. *Phys. Rev. Lett.* **126**, 098302, DOI: 10.1103/PhysRevLett.126.098302 (2021).
7. Raissi, M., Perdikaris, P. & Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707, DOI: https://doi.org/10.1016/j.jcp.2018.10.045 (2019).
8. Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* **3**, 127–149, DOI: 10.1016/j.cosrev.2009.03.005 (2009).
9. Pathak, J. et al. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos* 041101, DOI: 10.1063/1.5028373 (2018).
10. Arcomano, T. *et al.* A hybrid approach to atmospheric modeling that combines machine learning with a physics-based numerical model. *J. Adv. Model. Earth Syst.* **14**, e2021MS002712, DOI: https://doi.org/10.1029/2021MS002712 (2022). E2021MS002712 2021MS002712.
11. Montavon, G., Binder, A., Lapuschkin, S., Samek, W. & Müller, K.-R. Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. *Lect. Notes Comput. Sci.* 193–209, DOI: 10.1007/978-3-030-28954-6_10 (2019).
12. Mamalakis, A., Ebert-Uphoff, I. & Barnes, E. A. xxAI - Beyond Explainable AI, International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers. *Lect. Notes Comput. Sci.* 315–339, DOI: 10.1007/978-3-031-04083-2_16 (2022).
13. Subel, A., Guan, Y., Chattopadhyay, A. & Hassanzadeh, P. Explaining the physics of transfer learning in data-driven turbulence modeling. *PNAS nexus* **2**, pgad015, DOI: 10.1093/pnasnexus/pgad015 (2023).
14. Bommer, P., Kretschmer, M., Hedström, A., Bareeva, D. & Höhne, M. M. Finding the right XAI method - A guide for the evaluation and ranking of explainable AI methods in climate science. *CoRR* **abs/2303.00652**, DOI: 10.48550/ARXIV.2303.00652 (2023).