Neural network approach to differentiation of sampled multivariate functions

Frédéric Latrémolière[1], Sadananda Narayanappa[2], and Petr Vojtěchovský[1]

[1] University of Denver, Denver, Colorado, USA
`frederic@math.du.edu, petr@math.du.edu`
[2] Lockheed Martin Space Littleton Colorado, USA
`sadananda.narayanappa@lmco.com`

Abstract

We show how to train neural networks to differentiate a multivariate function F from sample values $(x, F(x))$. The resulting Jacobian matrix (of partial derivatives) can be used to peek into the structure of the underlying data. Unlike in other approaches, the neural network itself is never differentiated. We provide a rigorous proof of convergence under reasonable assumptions on the function, sample points and the neural network. The full preprint can be seen at https://arxiv.org/abs/2204.00523.

1 Introduction

We propose to use neural networks to approximate the Jacobian matrix J of a multivariate function F, where in the training of the neural network we use only a finite sample of input-output pairs $(x, F(x))$. Crucially, no additional information about F or J enters into the training and thus, in essence, the method proposed here differentiates a multivariate function F solely based on a cloud of sample points.

The algorithm is based on two main ideas:

- a loss function that utilizes a linear approximation of the sampled function F in terms of the sought-after Jacobian matrix,
- a nearest neighbor search in preparation of the training data from samples.

The neural network is not differentiated during or at the conclusion of the training.

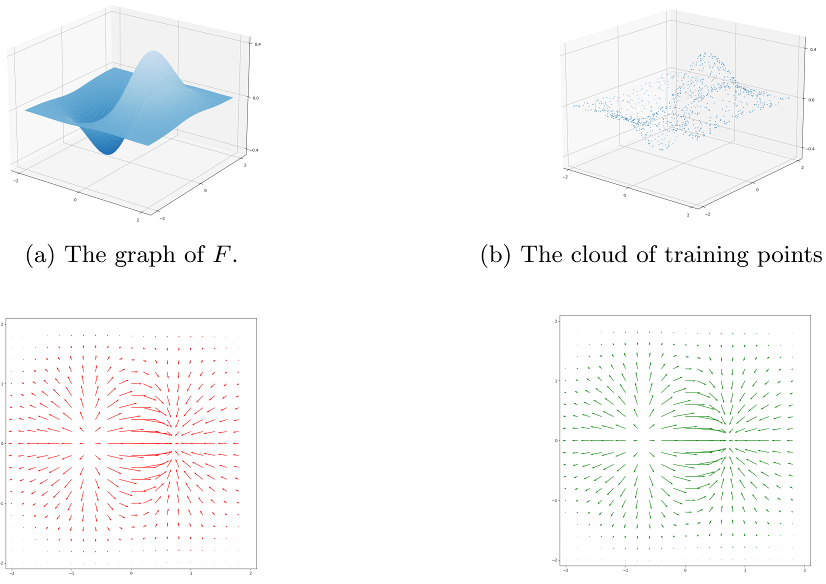
In general, to estimate J from sample points of F is a difficult mathematical problem. Since the Jacobian has numerous applications in mathematics, science and engineering, the ability to estimate it by any means, for instance by a neural network, is valuable in its own right.

Of particular importance here is the fact that the Jacobian can be invoked to detect relations, or lack thereof, between the input variables and output values of the unknown function F. While neural networks are a powerful tool for nonlinear regression, the resulting interpolating function is notoriously a black-box, revealing little structural information about the sampled function F. By taking advantage of the estimated Jacobian matrix of F, we can start peering inside F. We therefore expect that the ideas presented here will contribute to new approaches for the training of neural networks designed to reveal structural information about sampled data.

2 A visual example, error estimates and proof of convergence

Consider the function $F : \mathbb{R}^2 \to \mathbb{R}$, $(x, y) \mapsto x \exp(-x^2 - y^2)$ on the domain $(-2, 2)^2$. This function is not known to the algorithm for the purposes of training the Jacobian matrix estimator $\widehat{J}$, nevertheless it is known to us and we can therefore visualize it, calculate its Jacobian matrix J by standard symbolic differentiation, and visualize the Jacobian matrix as well.

The graph of F is given in Figure 1(a). Its Jacobian matrix J is a function $\mathbb{R}^2 \to \mathbb{R}^2$ and we plot it as a vector field on a regular grid of 20×20 points contained in $(-2, 2)^2$. This is done in Figure 1(c), where the vectors are automatically scaled to improve legibility. The input of the algorithm is a cloud of $N = 10^6$ sample points $\{(x, F(x)) : x \in X\}$, where the set $X \subseteq (-2, 2)^2$ is generated randomly. Such a cloud is visualized in Figure 1(b), except that only 10^3 points are plotted in the figure to improve legibility. The resulting Jacobian matrix estimator $\widehat{J}$ is visualized as a vector field in Figure 1(d).



(a) The graph of F.

(b) The cloud of training points for $\widehat{J}$.

(c) The gradient J (scaled).

(d) The estimated gradient $\widehat{J}$ (scaled).

Figure 1: A visual comparison of the Jacobian J of F and the trained Jacobian estimator $\widehat{J}$.

The algorithm for training of $\widehat{J}$ was implemented in TensorFlow.

In many validation tests, the average relative error between the actual Jacobian J and the estimated Jacobian $\widehat{J}$ was typically less than 3 percent, and it remained in single percentage digitis even for small sample sets (e.g., $N = 10^3$) and for higher dimensional functions (e.g., $\mathbb{R}^4 \to \mathbb{R}^4$).

We prove rigorously that the Jacobian estimator $\widehat{J}$ converges uniformly to the actual Jacobian J:

Theorem 2.1. *Let* $\varepsilon > 0$*, let* U *be an open bounded subset of* $\mathbb{R}^d$*,* $F : U \to \mathbb{R}^c$ *differentiable and* J *the Jacobian matrix of* F*. Suppose that certain assumptions are satisfied. (See the paper for all details on the assumptions.) Then there exists a constant* C *such that*

$$\sup_{x \in U} \left\| \widehat{J}(x) - J(x) \right\|_{c \times d} \leq C\varepsilon.$$

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015. Software available from `tensorflow.org`.

[2] J. Berner, D. Elbrächter, P. Grohs and A. Jentzen, *Towards a regularity theory for ReLU networks – chain rule and global error estimates*, 2019 13th International conference on Sampling Theory and Applications (SampTA), 2019, pp. 1–5, doi: 10.1109/SampTA45681.2019.9031005.

[3] Arthur Bryson, *A gradient method for optimizing multi-stage allocation processes*, Proceedings of the Harvard Univ. Symposium on digital computers and their applications, pp. 3–6, April 1961. Harvard University Press.

[4] John Conway, *A course in functional analysis*, 2nd edition. *Graduate Texts in Mathematics, 96.* Springer-Verlag, New York, 1990, xvi+399.

[5] Stuart Dreyfus, *The numerical solution of variational problems*, Journal of Mathematical Analysis and Applications, **5** (**1**), pp. 30–45. doi:10.1016/0022-247x(62)90004-5

[6] Michael C. Fu, *Gradient Estimation*, in Handbooks in Operations Research and Management Science, vol. **13** "Simulation", pp. 575–616, North-Holland, 2006.

[7] Kunihiko Fukushima, *Visual feature extraction by a multilayered network of analog threshold elements*, IEEE Transactions on Systems Science and Cybernetics **5** (**4**) (1969), pp. 322–333. doi:10.1109/TSSC.1969.300225.

[8] C.R. Harris, K.J. Millman, S.J. van der Walt et al, *Array programming with NumPy*, Nature **585** (2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[9] Xing He, Lei Chu, Robert Qiu, Qian Ai and Wentao Huang, *Data-driven Estimation of the Power Flow Jacobian Matrix in High Dimensional Space*, arxiv.org/abs/1902.06211

[10] Henry Kelley, *Gradient theory of optimal flight paths*, ARS Journal. **30** (**10**), pp. 947–954. doi:10.2514/8.5282

[11] I.E. Lagaris, A. Likas and D.I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, in IEEE Transactions on Neural Networks, vol. **9**, no. **5**, pp. 987–1000, Sept. 1998, doi: 10.1109/72.712178.

[12] Dougal Maclaurin, David Duvenaud and Ryan P. Adams, *Autograd: Effortless gradients in numpy*, in ICML 2015 AutoML Workshop **238** (2015), 5 pages.

[13] K. Rudd, G.D. Muro and S. Ferrari, *A Constrained Backpropagation Approach for the Adaptive Solution of Partial Differential Equations*, in IEEE Transactions on Neural Networks and Learning Systems, vol. **25**, no.**3**, pp. 571–584, March 2014, doi: 10.1109/TNNLS.2013.2277601.

[14] Walter Rudin, *Principles of mathematical analysis*, 3rd edition, International Series in Pure and Applied Mathematics. *McGraw-Hill Book Co., New York-Auckland-Düsseldorf*, 1976, x+342 pp.

[15] David Rumelhart, Geoffrey Hinton, Ronald Williams, *8. Learning Internal Representations by Error Propagation* in *Rumelhart, David E.; McClelland, James L. (eds.). Parallel Distributed Processing : Explorations in the Microstructure of Cognition. Vol. 1 : Foundations.* MIT Press. ISBN 0-262-18120-7.

[16] Carl Runge, *Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten*, Zeitschrift für Mathematik und Physik **46** (1901), pp. 224–243.

[17] Guanhua Wang and Jeffrey A. Fessler, *Efficient approximation of Jacobian matrices involving a non-uniform fast Fourier transform (NUFFT)*, arxiv.org/abs/2111.02912.