Deep Learning 2021 - Project Proposal

December 11, 2021

Sven Gregorio: svengr@student.ethz.ch, **Ahmet Solak:** asolak@student.ethz.ch,
Nils Ebeling: ebelingn@student.ethz.ch, **Mathias Ruoss:** mruoss@student.ethz.ch

Polynomial Feature Learning

1 Idea

Although it has been shown that Artificial Neural Networks are universal approximators [5], shallow neural networks may require exponentially many neurons to learn arbitrary functions [2], while deep neural networks often require long training times [1]. A well known result from linear regression is that polynomial relationships between features can contain useful information about the predicted variable [4]. We want to explore a method to learn polynomial features in neural networks, in the hope that it leads to simple neural networks which are easy to train and at least as powerful as their much deeper equivalents.

2 Method

We observe that polynomial functions can be expressed as neural networks if we use the natural logarithm and exponential function as activation functions. This can be seen as follows, consider a general polynomial function with n variables and m monomials:

$$P_{nm}[\mathbf{x}] = a_1 x_1^{e_{11}} x_2^{e_{21}} \dots x_n^{e_{n1}} + \dots + a_m x_1^{e_{1m}} x_2^{e_{2m}} \dots x_n^{e_{nm}}$$

A monomial can be described as a neuron taking logarithms of the variables as inputs and with an exponential activation:

$$x_1^{e_{1k}} x_2^{e_{2k}} \dots x_n^{e_{nk}} = \exp\left(\log x_1^{e_{1k}} x_2^{e_{2k}} \dots x_n^{e_{nk}}\right) = \exp\left(e_{1k} \log x_1 + e_{2k} \log x_2 \dots + e_{nk} \log x_n\right).$$

The polynomial can be seen as a linear combination of the different monomials. Thus, a layer with logarithmic activation followed by a layer with exponential activation can represent any polynomial function.
A neural network encoded as described above is differentiable in both the coefficient and exponents of the described general polynomial, so it could in theory learn any polynomial through gradient descent.

3 Challenges of the described approach

The transformation shown in the previous section has some properties which make it hard to use in practice. From a mathematical perspective, the logarithm on the real numbers is only defined for positive arguments, while neural networks generally can contain any real number. Additionally, if any one of the input variables approaches zero, it can make a polynomial as described above vanish, since all monomials include all variables. Also, exponents learned through an unconstrained optimization algorithm may not be integer valued, and may even be negative. Although this need not be a problem, we would have to explore to which extent we need to constrain the learned exponents.
There are possible solutions to the challenges described, but their number is too great to describe all of them in this proposal. The aim of our project is to explore some of these solutions and evaluate them in both synthetic and real settings.

4 Dataset

Since we have no guarantee that the proposed method will actually succeed, we intend to first use synthetic datasets to explore its potential. Once we have a good grasp on the the properties of the learned polynomials, we will apply the knowledge we acquired on well known datasets like MNIST [3] and CIFAR-10 [6].

5 Baseline

We will compare our method with other machine learning approaches, ranging from linear and polynomial regression, to deep neural networks, depending on the problem at hand.

References

[1] Tom B. Brown et al. “Language Models are Few-Shot Learners”. In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.

[2] Nadav Cohen, Or Sharir, and Amnon Shashua. *On the Expressive Power of Deep Learning: A Tensor Analysis*. 2016. arXiv: 1509.05009 [cs.NE].

[3] Li Deng. “The mnist database of handwritten digit images for machine learning research”. In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[4] J.D Gergonne. “The application of the method of least squares to the interpolation of sequences”. In: *Historia Mathematica* 1.4 (1974), pp. 439–447. ISSN: 0315-0860. DOI: https://doi.org/10.1016/0315-0860(74)90034-2. URL: https://www.sciencedirect.com/science/article/pii/0315086074900342.

[5] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[6] Alex Krizhevsky. “Learning Multiple Layers of Features from Tiny Images”. In: *University of Toronto* (May 2012).