ReLEx: Regularisation for Linear Extrapolation in Neural Networks with Rectified Linear Units

Enrico Lopedoto and Tillman Weyde

Department of Computer Science
City, University of London

Abstract. Despite the great success of neural networks in recent years, they are not providing useful extrapolation. In regression tasks, the popular Rectified Linear Units do enable unbounded linear extrapolation by neural networks, but their extrapolation behaviour varies widely and is largely independent of the training data. Our goal is instead to continue the local linear trend at the margin of the training data. Here we introduce ReLEx, a regularising method composed of a set of loss terms design to achieve this goal and reduce the variance of the extrapolation. We present a ReLEx implementation for single input, single output, and single hidden layer feed-forward networks. Our results demonstrate that ReLEx has little cost in terms of standard learning, i.e. interpolation, but enables controlled univariate linear extrapolation with ReLU neural networks.

Keywords: Neural Networks · Regression · Regularisation · Extrapolation.

1 Introduction

Neural Networks (NN) are very successful for many applications in artificial intelligence and the rectified linear function used in rectified linear units (ReLU) has become the most popular activation function. However, neural networks with ReLUs produce widely varying extrapolation behaviour that is determined more by the random initialisation of the network weights than by the training data.

Our goal in this study is to improve the extrapolation behaviour of NNs with ReLUs for regression tasks in three ways:

1. the extrapolation should be mainly defined by the data and not the random initialisation of the NN;
2. the extrapolation should continue the local linear trend with more influence from training data points closer to the margin;
3. the non-linearities should not be outside the training data range.

To realise these goals we develop, implement and evaluate ReLEx, a regularisation method to control extrapolation. In this study, we focus specifically on univariate regression with a single input and output and a single hidden layer, and do not address multi-dimensional settings.

2 Related Work

Neural Networks (NN) are universal function approximators [1,5] on a compact interval. While it has often been observed that neural networks interpolate very effectively between data points, it has been found in many studies that neural networks do not extrapolate well, where extrapolation is generally understood as the application of the trained model to data that is in some sense outside the range of the training data [6,8]. This can even affect very simple functions like the identity, which NNs do not generalise well to unseen data [11]. Linguistic structures are an example that recently attracted attention. It has been shown that NN do not generalise between different words [3]. This problem is being addressed in current research with more complex NN architectures [4].

For numeric extrapolation, there have recently been several studies that aim at learning functions that are more akin to those used physics by including multiplication, linear and periodic functions as activation functions [7,10]. These networks can produce good extrapolation if they can express the true function and the training finds suitable weights. However, this is not the case in general so that the extrapolation behaviour is unpredictable.

In this study, we choose a simpler approach, which uses a standard network architecture and activation function, but aims to control a linear extrapolation such that it matches the data. Linear extrapolations are often a simplification, but they are often useful, e.g. in LIME (Local Interpretable Model-Agnostic Explanations), which is widely used for explaining the behaviour of complex models [9].

3 Model

We use a simple fully connected feed-forward network that has a single hidden layer of N ReLUs, a single input neuron and a single linear output neuron. The network output for an input x_i is $\hat{y}_i = b_o + \sum_{n=1}^{N} w_{on} \cdot ReLU(x_i \cdot w_n + b_n)$, where b_o is the bias of the output neuron and w_{on} and w_n are the outgoing and incoming weights, respectively, of the n^{th} hidden neuron and b_n is its bias. We use stochastic gradient descent to train the network.

The neurons are rectified linear units, that use a rectified linear function as the activation function: $ReLU(x) = x \,\text{if}\, x > 0,\, 0 \,\text{otherwise}$. In the constant part ($x < 0$) the gradient of the activation is always 0. If the activation of the ReLU is in the constant part for all inputs, the ReLU does not contribute to the output of the network and this is called a 'dying ReLU' [2].

We are particularly interested in the *0-Points*, where the non-linearities of the hidden neurons are: the input values x_{k0} that lead to input 0 for the activation function of hidden neuron k: $0 = w_k \cdot x_{k0} + b_k$, such that $x_{k0} = -b_k/w_k$.

4 ReLEx Loss Definitions

We control the extrapolation behaviour of the neural network with additional loss terms that achieve our design goals in the learning process. The losses are Cen-

tripetal (CP), Mutually Repellent (MR), Weight Orientation (WO) and Weight Sign (WS) and are defined below in more details. The final loss to be minimised, including the sum of squared errors, is therefore:

$$\mathcal{L} = \sum(\hat{y} - y)^2 + \theta_{CP}\,\mathcal{L}_{CP} + \theta_{MR}\,\mathcal{L}_{MR} + \theta_{WO}\,\mathcal{L}_{WO} + \theta_{WS}\,\mathcal{L}_{WS}, \qquad (1)$$

with $\theta_{\mathcal{X}}$ being the weighting factors for each additional loss term $\mathcal{L}_{\mathcal{X}}$.

Centripetal Loss $\mathcal{L}_{CP}$, aims to move the 0-points inside the training range to fulfil design goals 1 and 3. It is defined as the sum of the squared distances between the 0-points and the the data points:

$$\mathcal{L}_{CP} = \sum_{k}^{N}\sum_{i}^{K}(x_{k0} - x_i)^2, \qquad (2)$$

where N is the number of hidden neurons and K is the number of data points.

Mutual Repellent Loss $\mathcal{L}_{MR}$, is designed to fulfil design goal 2 by avoiding the concentration of 0-points around the data mean, which can be the effect of $\mathcal{L}_{CP}$. This loss term aims to equally distribute the 0-points by penalising pairs of 0-points with a small distance. $\mathcal{L}_{MR}$ is the sum of the inverted pairwise distances between the positions of the 0-points with a small constant ε added to avoid division by zero:

$$\mathcal{L}_{MR} = \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{(x_{i0} - x_{j0})^2 + \varepsilon} \qquad (3)$$

Weight Orientation Loss $\mathcal{L}_{WO}$ also relates to design goal 2. The data points that are closer to the margin of the data range should influence the extrapolation beyond that margin more strongly. This can be achieved by the linear parts of the ReLUs covering more of the data points that are close to the margin. Intuitively speaking, we encourage the linear part of the ReLU to point outwards by penalising when a linear part of a ReLU covers more than half the data range:

$$\mathcal{L}_{WO} = -\sum_{i=1}^{N} ReLU((x_{k0} - \bar{x}) \cdot w_i + \varepsilon), \qquad (4)$$

where $\bar{x}$ is the data mean and ε is a small amount of linear part beyond the data mean that we do not penalise.

Weight Sign Loss L_{WS} prevents a degenerate condition and encourages an equal distribution of weight signs so that linear parts of ReLUs are pointing to both directions of extrapolation:

$$L_{WS} = \left(\sum_{i=1}^{N} w_i\right)^2. \qquad (5)$$

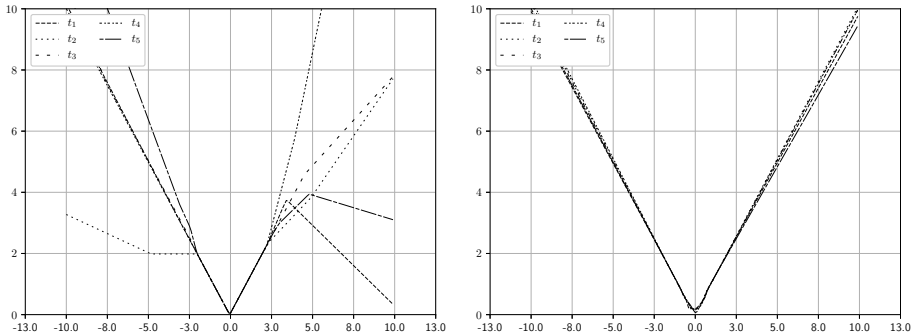

Fig. 1: Comparison of extrapolation after standard (left) and ReLEx training (right) with 5 models each. Input on x axis, output on y axis.

5 Experiments

5.1 Settings and Metrics

We use five functions to generate data: identity, absolute, scaled squared, sigmoid and sine $(x, |x|, x^2/5, 1/(1+e^{-x}), sin(x))$. We sample on 200 equidistant points for the training data-set and another 200 points at random for the interpolation test set from the range $[-2, 2]$. The extrapolation data range is $[-10, 10]$ with another 200 points. To measure extrapolation performance, we propose several metrics as there is no natural target when we are not assuming a true underlying linear function:

MSE_{int} measures the interpolation error on test data that was randomly sampled from the training data range.

σ_{ES} measures the consistency of the extrapolation as the standard deviation of the slope of the extrapolations as determined by a linear regression over the NN predictions on the test set on ranges $x \in [-10, -2]$ and $[2, 10]$.

MSE_{ex-tan} measures extrapolation accuracy as the MSE relative to the tangent of the generating function at the margins of the training data range.

MSE_{ex-wlr} measures accuracy against a more realistic extrapolation reference: a linear regression on the training data with higher weights on the data points closer to the margin of the data range. We use this function for weighting $w_{x,\tau} = x^{(\tau-1)}(1-x)^{(\tau-1)}$, where x is the position of the data point in the data range and τ is a free parameter.

5.2 Results

Preliminary experiments confirmed that the $\mathcal{L}_{CP}$ is effective at concentrating the 0-points. θ_{CP} was set to 0.03 throughout, which was determined as an effective values in most contexts. One issue that we observed was that for high θ_{CP}, values all 0-points would cluster around the data mean. This prevented a good

Method	$f(\cdot)$	MSE_{int}	MSE_{ex-tan}	MSE_{ex-wlr}	σ_{ES}
Std	x	0.00	18.41	18.45	1.18
ReLEx	x	0.03	0.02	0.02	0.02
Std	$\|x\|$	0.00	6.32	7.13	1.31
ReLEx	$\|x\|$	0.18	0.01	0.45	1.00
Std	$x^2/5$	0.02	14.63	14.06	12.92
ReLEx	$x^2/5$	0.03	0.07	0.04	10.50
Std	$1/(1+e^{-x})$	0.01	4.55	3.97	7.83
ReLEx	$1/(1+e^{-x})$	0.03	0.11	0.02	2.48
Std	$\sin(x)$	12.65	41.71	53.14	1.36
ReLEx	$\sin(x)$	0.43	1.82	34.78	0.02

Table 1: The effect of applying all ReLEx losses (CP, MR, WO and WS).

fit and extrapolation in some cases, which is the reason for introducing $\mathcal{L}_{MR}$. By choosing a suitable ratio between θ_{MR} and θ_{CP}, we can adjust the 0-point spread to the data range.

Figure 1 show results from standard NN and ReLEx networks and makes clear how ReLEx reduces extrapolation variability and leads to a good continuation of the local linear trend at the margins. When comparing to the weighted linear extrapolation in the MSE_{ex-wlr} loss, we found that τ values above 0.5, indicating higher weights for points close to the margin, gave better results. We used a final τ value of 0.8, which generally leads to a good fit.

Table 1 shows the metrics for ReLEx in comparison with standard network training. We can see that the MSE_{int} is in most cases not affected much by ReLEx, except for an increase for the $|x|$ function and a decrease for the $\sin(x)$. The extrapolation error against the tangents (MSE_{ex-tan}) is in all cases greatly reduced, as is the error against the WLR extrapolation (MSE_{ex-wlr}). Both extrapolation error measures are higher for the $\sin(x)$ function compared to other functions, while the standard deviation of the slope (σ_{ES}) is low, indicating that for the $\sin(x)$ function the ReLEx NN extrapolation behaves differently than for the other functions. In terms of standard deviation of the extrapolation slope, the quadratic function ($x^2/5$) shows much higher spread, even with ReLEx.

$\mathcal{L}_{WO}$ and $\mathcal{L}_{WS}$ can efficiently be calculated in $O(N)$ time. However, the computational complexity of calculating $\mathcal{L}_{CP}$ and $\mathcal{L}_{MR}$ is $O(N \cdot K)$ and $O(N^2)$, respectively. These may be computationally too expensive for large models and data-sets. There are alternative variants with $O(N)$ complexity, such as replacing both with a small penalty on distance to the mean, or fixed boundaries for the 0-points, but they require an extra pass through the data before training.

6 Conclusions

The ReLU function offers the possibility of performing linear extrapolation, which standard bounded activation functions do not. However, they show high

variability and little correlation with the data when trained, as standard, by minimising just the error against the data. We proposed ReLEx, a set of loss terms that provide regularisation for the behaviour of a ReLU network, such that it produces linear extrapolations beyond the range of the training data that are consistent and continue the local linear trend of the data at the margins.

The accuracy of the extrapolation has no single definition as there is no pre-defined target, but measured against the tangent and a weighted linear regression, ReLEx shows large improvements over a standard network. The ReLEx loss terms can be directly integrated into a standard neural network framework, as we have done with our implementation in PyTorch.[1]

The computational complexity of the loss functions can become relevant for large networks and large data-sets, but had little impact in our experiments. Future work will include the exploration of more efficient ReLEx variants, the generalisation to multi-dimensional inputs and outputs and multiple layers, as well as experiments with noisy and real-world data.

References

1. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems **2**(4), 303–314 (1989)
2. Douglas, S.C., Yu, J.: Why ReLU units sometimes die: Analysis of single-unit error backpropagation in neural networks. In: 52nd Asilomar Conference on Signals, Systems, and Computers. pp. 864–868. IEEE (2018)
3. Lake, B., Baroni, M.: Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In: PMLR. vol. 80, pp. 2873–2882 (2018)
4. Lake, B.M.: Compositional generalization through meta sequence-to-sequence learning. In: NeurIPS. pp. 9791–9801 (2019)
5. Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Networks **6**(6), 861–867 (1993)
6. Marcus, G.: Deep learning: A critical appraisal. CoRR abs/1801.00631 (2018)
7. Martius, G., Lampert, C.H.: Extrapolation and learning equations. In: ICLR, Workshop Track Proceedings (2017)
8. Mitchell, J., Minervini, P., Stenetorp, P., Riedel, S.: Extrapolation in NLP. In: Proceedings of the Workshop on Generalization in the Age of Deep Learning (2018)
9. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why Should I Trust You?": Explaining the predictions of any classifier. In: ACM KDD. p. 1135–1144 (2016)
10. Trask, A., Hill, F., Reed, S.E., Rae, J., Dyer, C., Blunsom, P.: Neural arithmetic logic units. In: NeurIPS. pp. 8035–8044 (2018)
11. Weyde, T., Kopparti, R.M.: Feed-forward neural networks need inductive bias to learn equality relations. In: NeurIPS Workshop on Relational Representation Learning (2018)

[1] The source code and more supporting material can be found at `https://github.com/EnricoLope/PhD_ReLEx`.