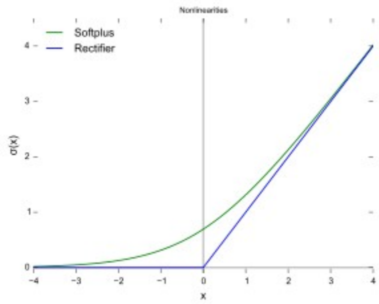
WIKIPEDIA

Rectifier (neural networks)

In the context of artificial neural networks, the **rectifier** is an activation function defined as the positive part of its argument:

$$f(x) = x^+ = \max(0, x),$$

where x is the input to a neuron. This is also known as a ramp function and is analogous to half-wave rectification in electrical engineering. This activation function was first introduced to a dynamical network by Hahnloser et al. in 2000 with strong biological motivations and mathematical justifications.[1][2] It has been demonstrated for the first time in 2011 to enable better training of deeper networks,[3] compared to the widely-used activation functions prior to 2011, e.g., the logistic sigmoid (which is inspired by probability theory; see logistic regression) and its more practical[4] counterpart, the hyperbolic tangent. The rectifier is, as of 2018, the most popular activation function for deep neural networks.[5][6]



Plot of the rectifier (blue) and softplus (green) functions near $x = 0$

A unit employing the rectifier is also called a **rectified linear unit** (**ReLU**).[7]

Rectified linear units find applications in computer vision[3] and speech recognition[8][9] using deep neural nets.

Contents

Softplus

A smooth approximation to the rectifier is the analytic function

$$f(x) = \log(1 + e^x),$$

which is called the **softplus**[10][3] or **SmoothReLU** function.[11] The derivative of softplus is $f'(x) = \dfrac{e^x}{1+e^x} = \dfrac{1}{1+e^{-x}}$, the logistic function. The logistic function is a smooth approximation of the derivative of the rectifier, the Heaviside step function.

The multivariable generalization of single-variable softplus is the LogSumExp with the first argument set to zero:

$$LSE_0^+(x_1, \dots, x_n) := LSE(0, x_1, \dots, x_n) = \log(1 + e^{x_1} + \dots + e^{x_n}).$$

The LogSumExp function itself is:

$$\mathrm{LSE}(x_1, \dots, x_n) = \log(e^{x_1} + \dots + e^{x_n}),$$

and its gradient is the softmax; the softmax with the first argument set to zero is the multivariable generalization of the logistic function. Both LogSumExp and softmax are used in machine learning.

Variants

Noisy ReLUs

Rectified linear units can be extended to include Gaussian noise, making them noisy ReLUs, giving[7]

$$f(x) = \max(0, x + Y), \text{ with } Y \sim \mathcal{N}(0, \sigma(x))$$

Noisy ReLUs have been used with some success in restricted Boltzmann machines for computer vision tasks.[7]

Leaky ReLUs

Leaky ReLUs allow a small, positive gradient when the unit is not active.[9]

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{otherwise} \end{cases}$$

Parametric ReLUs

Parametric ReLUs (PReLUs) take this idea further by making the coefficient of leakage into a parameter that is learned along with the other neural network parameters.[12]

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases}$$

Note that for $a \leq 1$, this is equivalent to

$$f(x) = \max(x, ax)$$

and thus has a relation to "maxout" networks.[12]

ELUs

Exponential linear units try to make the mean activations closer to zero which speeds up learning. It has been shown that ELUs can obtain higher classification accuracy than ReLUs.[13]

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ a(e^x - 1) & \text{otherwise} \end{cases}$$

a is a hyper-parameter to be tuned and $a \geq 0$ is a constraint.

Advantages

- Biological plausibility: One-sided, compared to the antisymmetry of tanh.
- Sparse activation: For example, in a randomly initialized network, only about 50% of hidden units are activated (having a non-zero output).
- Better gradient propagation: Fewer vanishing gradient problems compared to sigmoidal activation functions that saturate in both directions.[3]
- Efficient computation: Only comparison, addition and multiplication.
- Scale-invariant: $\max(0, ax) = a\max(0, x)$ for $a \geq 0$.

Rectifying activation functions were used to separate specific excitation and unspecific inhibition in the Neural Abstraction

Pyramid, which was trained in a supervised way to learn several computer vision tasks.[14] In 2011,[3] the use of the rectifier as a non-linearity has been shown to enable training deep supervised neural networks without requiring unsupervised pre-training. Rectified linear units, compared to sigmoid function or similar activation functions, allow for faster and effective training of deep neural architectures on large and complex datasets.

Potential problems

- Non-differentiable at zero; however it is differentiable anywhere else, and a value of 0 or 1 can be chosen arbitrarily to fill the point where the input is 0.
- Non-zero centered
- Unbounded
- Dying ReLU problem: ReLU neurons can sometimes be pushed into states in which they become inactive for essentially all inputs. In this state, no gradients flow backward through the neuron, and so the neuron becomes stuck in a perpetually inactive state and "dies." This is a form of the vanishing gradient problem. In some cases, large numbers of neurons in a network can become stuck in dead states, effectively decreasing the model capacity. This problem typically arises when the learning rate is set too high. It may be mitigated by using Leaky ReLUs instead, which assign a small positive slope to the left of x = 0.

See also

- Softmax function
- Sigmoid function
- Tobit model

References

1. Hahnloser, R.; Sarpeshkar, R.; Mahowald, M. A.; Douglas, R. J.; Seung, H. S. (2000). "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit". *Nature*. **405**: 947–951. doi:10.1038/35016072 (https://doi.org/10.1038%2F35016072).
2. R Hahnloser, H.S. Seung (2001). *Permitted and Forbidden Sets in Symmetric Threshold-Linear Networks*. NIPS 2001.
3. Xavier Glorot, Antoine Bordes and Yoshua Bengio (2011). *Deep sparse rectifier neural networks* (http://jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf) (PDF). AISTATS. "Rectifier and softplus activation functions. The second one is a smooth version of the first."
4. Yann LeCun, Leon Bottou, Genevieve B. Orr and Klaus-Robert Müller (1998). "Efficient BackProp" (http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf) (PDF). In G. Orr and K. Müller. *Neural Networks: Tricks of the Trade*. Springer.
5. LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep learning". *Nature*. **521** (7553): 436–444. Bibcode:2015Natur.521..436L (http://adsabs.harvard.edu/abs/2015Natur.521..436L). doi:10.1038/nature14539 (https://doi.org/10.1038%2Fnature14539). PMID 26017442 (https://www.ncbi.nlm.nih.gov/pubmed/26017442).
6. Ramachandran, Prajit; Barret, Zoph; Quoc, V. Le (October 16, 2017). "Searching for Activation Functions". arXiv:1710.05941 (https://arxiv.org/abs/1710.05941) [cs.NE (https://arxiv.org/archive/cs.NE)].
7. Vinod Nair and Geoffrey Hinton (2010). *Rectified Linear Units Improve Restricted Boltzmann Machines* (https://www.cs.toronto.edu/~hinton/absps/reluICML.pdf) (PDF). ICML.
8. László Tóth (2013). *Phone Recognition with Deep Sparse Rectifier Neural Networks* (http://www.inf.u-szeged.hu/~tothl/pubs/ICASSP2013.pdf) (PDF). ICASSP.
9. Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng (2014). Rectifier Nonlinearities Improve Neural Network Acoustic Models (https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf)
10. Dugas, Charles; Bengio, Yoshua; Bélisle, François; Nadeau, Claude; Garcia, René (2000-01-01). "Incorporating second-order functional knowledge for better option pricing" (http://papers.nips.cc/paper/1920-incorporating-second-order-functional-knowledge-for-better-option-pricing.pdf) (PDF). *Proceedings of the 13th International Conference on Neural Information Processing Systems (NIPS'00)*. MIT Press: 451–457. "Since the sigmoid h has a positive first derivative, its primitive, which we call softplus, is convex"
11. "Smooth Rectifier Linear Unit (SmoothReLU) Forward Layer" (https://software.intel.com/sites/products/documentation/doclib/daal/daal-user-and-reference-guides/daal_prog_guide/GUID-FAC73B9B-A597-4F7D-A5C4-46707E4A92A0.htm). *Developer Guide for Intel® Data Analytics Acceleration Library*. 2017. Retrieved 2018-12-04.

12. He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2015). "Delving Deep into Rectifiers: Surpassing Human-Level Performance on Image *Net* Classification". arXiv:1502.01852 (https://arxiv.org/abs/1502.01852) [cs.CV (https://arxiv.org/archive/cs.CV)].
13. Clevert, Djork-Arné; Unterthiner, Thomas; Hochreiter, Sepp (2015). "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". arXiv:1511.07289 (https://arxiv.org/abs/1511.07289) [cs.LG (https://arxiv.org/archive/cs.LG)].
14. Behnke, Sven (2003). *Hierarchical Neural Networks for Image Interpretation* (https://www.researchgate.net/publication/220688219_Hierarchical_Neural_Networks_for_Image_Interpretation). Lecture Notes in Computer Science. **2766**. Springer. doi:10.1007/b11963 (https://doi.org/10.1007%2Fb11963).