Deep Learning **Ehsan Mahmoudi**
Assignment 1
Dr Bokaei

1 Activation Functions

1.1 tanh

Here is the formula for *tanh*:

$$\frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{1}$$

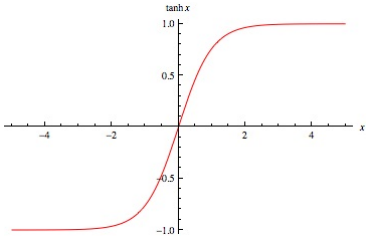

Figure 1: *tanh* Plot

And here is the derivative formula:

$$\frac{dtanh(x)}{dx} = sech^2(x) = \left(\frac{2}{e^x + e^{-x}}\right)^2 \tag{2}$$

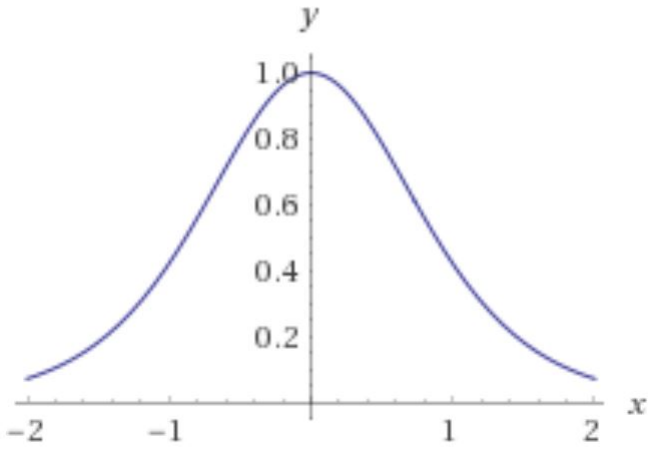

Figure 2: *tanh* Derivative Plot

1.2 Sigmoid

Here is the formula for σ:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

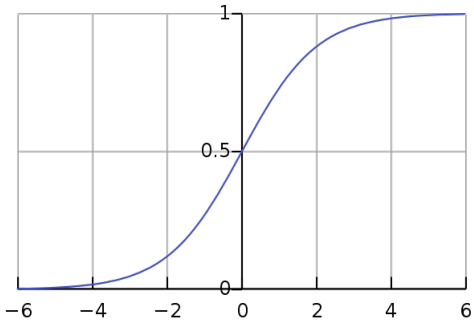

Figure 3: $\sigma(x)$ Plot

And here is the derivative formula:

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} \tag{4}$$

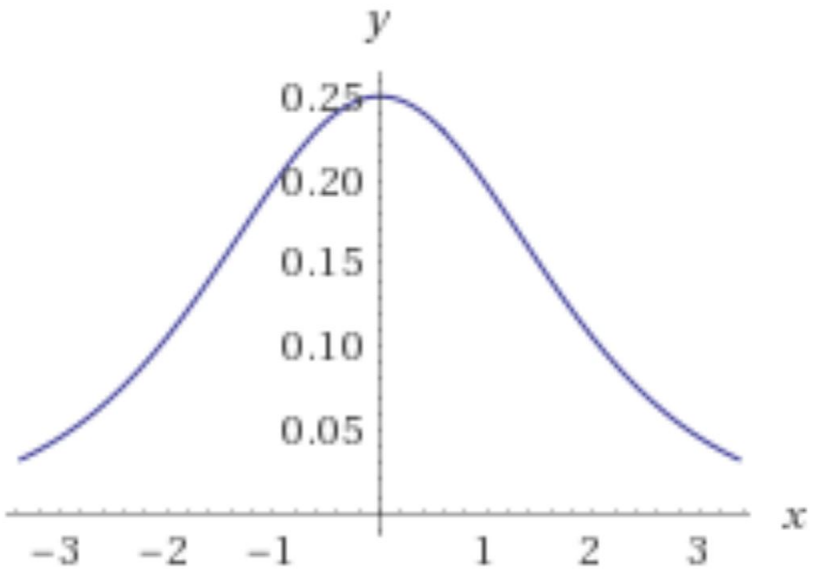

Figure 4: $\sigma(x)$ Derivative Plot

If we want to compare the above two functions σ and $tanh$ it seems $tanh$ is preferred because it is symmetric around 0. Having data of a layer (even hidden layers) to be around zero helps more with training. because of this also it is advised to have input data also normalized. Also having zero mean inputs to a layer makes more sense on regularizations that we apply on the weights, otherwise the layers with non-zero mean input will suffer more from regularizations. See [1]

2 ReLU

Here is the formula for ReLU activation function.

$$f(x) = x^+ = max(0, x) \tag{5}$$

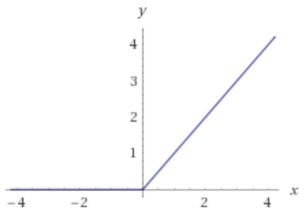

Figure 5: ReLU Plot

The problem with ReLU is that is not derivativable. And we need to calculate the derivative for backpropagation training. So we need to come up with a close function to be derivativable. There are two options available either calculate the gradient by numerical method or estimate the function using an analytic function such a soft-ReLU. On the other hand there are multiple advantages associated with ReLU. First of all it is more aligned with what we know from behaviour of biological neural systems. Also using ReLU activation function will result in more sparse representations of the data. This is useful specially when the network is deep. In practice also it is performing better on pre-training models such Boltzmann Machines or deep belief networks.

Also in ReLU as the output does not saturate, we have use some sort of regularization scheme to control the weight sizes. As one of the advantages of the ReLU unit is sparsity, it makes sense to use L_1 regularization which also promotes the sparsity. See [2]

References

[1] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50, London, UK, UK, 1998. Springer-Verlag. ISBN 3-540-65311-2. URL `http://dl.acm.org/citation.cfm?id=645754.668382`.

[2] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL `http://proceedings.mlr.press/v15/glorot11a.html`.