A Criteria Derived Activation Function

Bird, George george.bird@student.manchester.ac.uk
Polivoda, Maxim E. mxp19tkp@bangor.ac.uk

07 October 2020

Abstract

Artificial Neural Networks process information by applying transformations to data stored in manifolds which are embedded in high dimensional space. However, these manifolds may not be distributed evenly through all of available space. In fact it appears that current activation functions encourage such an uneven distribution that we must develop a set of criteria which activation functions should meet to ensure optimal manifold distribution on generalised tasks. We have also proposed several functions which meet this criteria.

1 Introduction

1.1 A Spatial Intuition for Neural Networks

Exactly why artificial neural networks process information in the way they do is poorly understood. However, a doctrine of imagining the network as a group of spatial transformations has had quite some success in demystifying the actions of artificial neural networks. This is encouraged by the extensive use of matrix and tensor mathematics, which are commonly understood as relating to spatial systems. This way of understanding networks spatially I will refer to as manifold theory. Manifold theory states that each datapoint, which is stored and processed by the network, is represented by a coordinate point within a hyperdimensional space or multiple spaces. When introducing many data points, from a real-world dataset, complex shapes known as manifolds form embedded in the space. These shapes are outlined by the positions of many of the aforementioned points. These manifold shapes are then manipulated by a variety of spatial transformations to yield a more useful and interpretable manifold. The dimensionality of the spaces is simply the number of neurons in a layer, where the neurons themselves are a set of perpendicular basis vectors whilst the information they currently hold reflects a coordinate for the current datapoint. Conversely any point in the space can have its position represented by a vector, where each component of the vector maps to a specific neuron's value. The spatial transformations are variable and change until they produce useful manifolds. This is the basic action of all neural networks, yet this analogy is most

evident for fully connected networks. It can, however, be expanded to account for all networks such as Convolutions and Long-Short Term Memory networks (LSTMs).

Although this doctrine is growing, it is still common for people to place too much emphasis on the behaviour of individual neurons rather than the collective behaviour of the network. I mention this as the basis for all activation functions to date seems to revolve around the behaviour of the individual neurons. I believe this has limited the effectiveness of many activation functions as group behaviour has been neglected in design choices. Therefore, we have compiled a set of criteria which activation functions should follow to ensure the neurons are treated as a collective.

1.2 Strict Function Criteria

Strict criteria are criteria which the functions must follow to be effectively optimised. We present arguments as to why each criterion is important using ideas taken from manifold theory.

1.2.1 Non-Linear

For a fully connected network with a linear activation function (or no activation function at all), the neural network consists of a series of linear transformations only. Thus the entire network's function could be summarised as a single fully connected layer, lacking the complex information processing which makes these networks particularly useful. Therefore, the first criteria is that the activation function must be non-linear. All widely used activation functions meet this criteria. It is effectively an essential condition if one wishes to build any basic network capable of recognising anything more than linear trends.

1.2.2 Equal input/output manifold dimensionality

Another criterion states that the dimensionality of the manifold should not change from the input to the output of the function. Removing a dimension results in the needless loss of information, and adding a dimension causes overhead. Once again, many existing functions conform to this criteria, however, softmax does not. Softmax reduces the dimensionality of the manifold.

Despite this, softmax has some features which make it desirable in specific use cases. This includes human-network interfaces such as the output layer of the network, as no further manifold transformations are taking place, so it is acceptable to lose some seemingly redundant data, and that the manifold needs to be distributed in a way that is comprehendible to humans, which is otherwise a sub-optimal arrangement. Also inside transformer networks it has a specific use case so our generalised activation function criteria does not apply. However, our next criterion rules out practically all commonly used activation functions. This is the necessity for a spherically symmetric space.

1.2.3 Spherically Symmetric

Any asymmetries result it uneven distribution of manifolds. This is simply due to certain regions of space being more preferable than others to store information. Thus, by introducing symmetries we attempt to make all space have even, and better suited probabilities to store manifolds. The more available space which manifolds can occupy, the more information the network can encode. To clarify we are not saying that activation functions which result in a finite output space are bad, but their finite bounds needs to be equidistant from the centre of the output space in all directions. Therefore, we need a spherically symmetric activation function; then there is no unfavourable direction.

By applying this condition the coordinate system used no longer influences the distribution of manifolds in space, thus it is not possible to figure out which direction the basis vectors (neurons) point just from observing the output space of the function. We can only determine the centre of the space. This makes the network behave as a more unified system rather than a collection of independent neurons. There is no longer anything special about individual neurons besides the way we notate the mathematical computations. No widely used activation functions I know of currently possess this property

Despite being an infinite space, both ReLU and Leaky-RelU collapse and scale only certain directions respectively. Thus, introducing favoured encoding directions. Sigmoid and Tanh operate on each points vector components consequently creating a hyper-cubic space. Hence the distance from the centre to the edge of the space is dependent on direction. As you can see this is a problem effecting many activation functions. Our networks so far have been performing sub-optimally with an effective loss in how much information can be encoded by a set of neurons. Many of the activation functions are plagued by us applying their transformations on an individual neuron basis rather than us imagining the network as a collective. The unfavoured directions may also make learning more troublesome for the network.

1.2.4 Projection

Finally there are many ways in which a spherically-symmetric space could be created for the purpose of an activation function. One could imagine taking a two-dimensional input space and wrapping distant regions in to an infinitely thin spiral pattern - creating a whirlpool-like spherically-symmetric finite output space. This function fulfils all our criteria so far, but would produce a very poorly performing network. The reason for this requires another look at manifold theory.

The neural network transforms the input manifold to express useful trends in the information. Since the network mostly uses linear transformations, except the activation function, to classify data, then these trends should be expressed linearly to allow the following layers in the network to mostly easily interpret them, especially considering the substantial use of linear algebra. We must also make the assumption that important trend-lines are most often origin-centric

or pass through the origin. In theory, if a point on the origin-centric vector changes sign, then the meaning of that point would be the direct opposite of the original point, analogous to an antonym. We shall refer to these important trend-lines as "meaning vectors" as they are often interpretable by humans. For example it is what Primary Component Analysis attempts to find. There could be a meaning vector for the amount of the colour red in the input data, where the negative sign may correspond to blue. Of course there are meaning-vectors between every permutations of all data points, however only some are important to the network and we shall assume that these ones are roughly origin-centric.

This origin-centricity is not a great assumption to make. However, non-linearities must be introduced to the function and due to the spherical symmetry then there is an enforced centre to the space. Thus, the non-linearity is isotropic around the centre to ensure symmetry. We hope this will cause the network to migrate its most important meaning-vectors to be origin centric. We mention this, as it is not possible using a non-linear function to keep meaning-vectors linear if they start anywhere in space. We can only keep vectors passing through a certain point linear and so we choose this to be the centre of the space.

We can reason that the following layers of the network will recognise these trend-lines and attempt to interpret them further: interpolating and extrapolating its findings. However if the space curves, then so does the meaning vector, thus any attempt at extrapolation fails as the network will assume it remains linear trend as it has observed near the origin. If the network accounts for the curving, it has dedicated more neurons to interpret the more complex representation this will also still extrapolate poorly as the network is using linear algebra to approximate this curve. This is needless, requiring more time for the network to learn and reduces the effective networks processing power on the input data. These problems can be avoided by ensuring that vectors through the origin remain linear.

Thus, any points on a line drawn through the origin should remain on that line before and after the transformation or more generally any points on a line drawn through the centre of the input space should all remain on another line drawn through the centre of the output space. It does not matter if all the input lines are rotated evenly about the centre during the activation function, however I see no purpose in doing so. In summary, the direction must not be dependent on the distance of a point from the centre of the space. It is also important to mention that the weights in fully connected networks are performing a transformation around the origin, this indicates that a fully connected network already is origin centric.

Overall by ensuring that all points on the centred line remain on another centred line after the transformation, then the 'meaning-vectors' passing through the origin should remain linear. The presence of linear meaning-vectors means that the network can more easily extrapolate trends and that these trends are more interpretable as the network does not need to performing extra processing to account for curved trend-lines. The network can then reason that along the trendline a property in the data gets stronger. In effect we are saying that the output space is a spherical projection of the input space.

1.2.5 Continuous

A continuous functions is our final strict criteria. However, the function does not need to be smooth. With a discontinuous function, potential gaps are introduced in to the manifold. There is no reason to do this and it just creates irregularities in the distribution of points in the manifold making it more difficult for the network to interpret.

Functions such as the step function violate many of our criteria, but particularly this discontinuity causes the manifold to lose large amounts of information as several regions of the manifold are collapsed to singular points. This should be avoided as it is effectively reducing the dimensionality of the manifold.

1.3 Relaxed Function Criteria

Relaxed criteria are criteria which are not as essential, and functions which do not meet these criteria may still work effectively in many situations. However, we feel that they should be avoided if possible.

1.3.1 Bounded Functions

Bounded Functions is referring to the output space of an activation function being finite. Thus as the radii of the input space goes to infinity, the output reaches some finite limiting radii from the centre. This is a relaxed criteria as we have seen much success using functions such as ReLU and leaky-ReLU in many applications. I feel these are very useful for specific use cases, but their infinite output space is not favourable in general purpose activation functions.

This is because outputs can reach huge values which may impede the function of later neurons as they become overloaded by such large values. This can be corrected for using a batch normalisation technique, however it is not always possible to use batch normalisations. Hence we introduce "bounded function" as a relaxed criterion.

1.3.2 Origin as the Centre of the Space

Our final relaxed criteria is most applicable to fully connected layers, due to the nature of linear algebra multiplications. Fully connected layers first multiply their inputs with a weight matrix before adding a bias vector. This corresponds to a transformation about the origin followed by a translation. It is important to note that all linear algebra transformation perform their transformation around the origin. Therefore, it would be sensible to give the network the oppurtunity to distribute the manifold around the origin, hence creating this final criteria.

There are many circumstances where this would not be ideal, such as an elementwise multiplication of two vectors intended to cause a "blocking-operation" such as those seen in Long-Short Term Memory Networks. Despite this, in general activation functions should have an output space centred around the origin if the output is to be further processed by fully connected layers.

As linear algebra multiplications transforms about the origin.

2 Criteria Fulfilling Functions

2.0.1 Geebs' Tanh Function

We first introduce the "Geebs' Tanh Function" which meets all of the strict and relaxed criteria (Eqn. 1). This is a modification of a standard Tanh formula. Instead of applying a tanh function elementwise, it instead applies tanh to the pythagorian distance of a point from the origin. Improvements should be observed by replacing most general uses of standard Tanh and Sigmoid within neural networks. However, such specific uses such as sigmoid providing a blocking factor in a network, for example the 'forget network' in an LSTM which uses a sigmoid function before an elementwise multiplication, should not be replaced as the required function's bounds are different from the Geeb's Tanh function. We provide an alternate equation to accomidate for this specific, albeit common, example later.

$$F\left(\bar{n}\right) = \hat{n} \tanh |\bar{n}| \tag{1}$$

Where $\bar{n}$ is the cartesian coordinates represented in a vector for the input data point. Shown below is the derivative of each element in the output vector in respect to each element of the input vector. This shows the function fulfills the non-linear criteria.

$$\frac{\partial F\left(\bar{n}\right)_i}{\partial \bar{n_j}} = \begin{cases} \frac{\left(|\bar{n}|^2 - |\bar{n}_i|^2\right) \tanh |\bar{n}|}{|\bar{n}|^3} + \frac{\bar{n}_i^2 sech^2 |\bar{n}|}{|\bar{n}|^3} & i = j \\ \frac{|\bar{n}| sech^2 |\bar{n}| - \tanh |\bar{n}|}{|\bar{n}|^3} \bar{n}_i \bar{n}_j & i \neq j \end{cases}$$

This function does not reduce the dimensionality of the space, shown by $\bar{n}'$ having the same number of elements as $\bar{n}$, and it does not reduce the dimensionality of the manifold which can be shown experimentally. All of our functions are constructed by two seperable equations, reflecting a hyper-spherical coordinate system used. There is the magnitude equation which is only dependent on the pythagorian distance of a point from the origin, consequently producing a spherically symmetric output space. The magnitude equation is given by $\tanh |\bar{n}|$. The other equation describes the direction points are distributed in, and this is done by projecting the space radially onto a hyper-spherical shell with magnitude one from the origin, thus ensuring the projection criteria. This direction equation simply converts any given vector representing a point from the origin to its corresponding unit vector. The direction equation is $\hat{n}$. Therefore, any point on a line through the origin remains on that line after the transformation. This function is also continuous, centred around the origin and bounded with a maximum radius of one from the origin. Geeb's Tanh function also happens to be smooth.

This function suffers from the same problem which Tanh and Sigmoid do. Its gradients tends to zero in the limit where the input point goes to an infinite distance from the origin. This incurs the vanishing gradients problem.

2.0.2 Maxim-ReLU Function

The second function we present is an analogue of the popular Leaky-ReLU function. Our Maxim-ReLU function meets all the strict criteria and all except one of the relaxed criteria. It fails to meet the relaxed bound function criteria. This is characteristic of a Leaky-ReLU function though, so it is acceptable in many applications. It applies a constant scaling given by α within a certain radius of the origin given by β. Consequently, we get a function similar to a typical Leaky-ReLU function except its scaling is applied within a certain range of the origin radially instead of the when the sign of a cartesian coordinate component is negative. Improvements should be made by replacing all uses of ReLU with Maxim-ReLU.

$$F(\bar{n}) = (\max(\alpha(\bar{n} - \beta), \bar{n} - \beta) + \alpha\beta) \times \hat{n} \tag{2}$$

Where $0 < \alpha \leq 1$ and $0 < \beta$ and $\bar{n}$ is the cartesian coordinates represented in a vector for the input data point. Shown below is the derivative of each element in the output vector in respect to each element of the input vector. This shows that the function fulfills the non-linear criteria.

$$\frac{\partial F(\bar{n})_i}{\partial \bar{n_j}} = \begin{cases} \alpha & i = j \cap |\bar{n}| < \beta \\ 0 & i \neq j \cap |\bar{n}| < \beta \\ \frac{-(\alpha-1)\bar{n}_i\bar{n}_j}{|\bar{n}|^3} & i = j \cap \beta < |\bar{n}| \\ \frac{(\alpha-1)\beta(|\bar{n}|-\bar{n}_i)}{|\bar{n}|^3} & i \neq j \cap \beta < |\bar{n}| \end{cases}$$

This function does not reduce the manifold's dimensionality. It can again be broken down into a magnitude equation $(\max(\alpha(\bar{n} - \beta), \bar{n} - \beta) + \alpha\beta)$ which is only dependent on the input data points' distance from origin and a direction $\hat{n}$ dependent only on the initial direction of the data point. Just as before, the direction equation is simply projecting all vectors onto a unit hyper-sphere. Due to these properties, our function meets both the spherical symmety and projection criteria. Its output space is also centred on the origin. However, this function is not bounded, just like a standard Leaky-ReLU, and is not smooth. Despite this, it does not suffer from gradients tending to zero as the input radius tends to infinity, therefore will not suffer badly from the vanishing gradients. This function should again provide improvements by replacing all cases of ReLU with Maxim-ReLU.

2.0.3 Hyper-Cubic Projection

We have now replaced two of the most commonly used activation functions, which covers most of the general uses of activation functions. However, as previously mentioned, there are some unique use cases in which our functions are not applicable. One of these use cases is a blocking gate in an LSTM, as this

requires a hyper-cubic output space rather than a hyper-spherical output space. This hyper-cubic output space must also be bounded between zero (blocked) and one (passed) for each component in a neuron-basis coordinate system. Of course this is incompatible with the spherical symmetry criteria, but since this is a unique use case and is widely used, we have proposed a projection equation to map an input space to a hyper-cubic space.

This equation projects the input space into a hyper-cubic space bounded between -1 and 1 for all neuron-basis components. A typical minimum function cannot be used here, as it collapses the space in a cubic fashion rather than a spherical fashion. In effect this function preserves the space between $-1 < \bar{n} < 1$ whilst collapsing all space outside this region into a hyper-cubic shell. This can be performed using a typical minimum function, however this does not satisfy the projection criteria, as a straight line through the origin gets collapsed onto a line on the shell rather than a single point. Thus, to preserve the linearity of meaning-vectors we have designed equation 3 to project points onto the shell radially instead of cubicly.

$$F\left(\bar{n}\right) = \min\left(\frac{1}{\max\left(\left|\bar{n}_1\right|, \left|\bar{n}_2\right|, ..., \left|\bar{n}_i\right|\right)}, 1\right) \times \bar{n} \tag{3}$$

As mentioned, this function's bounds are $-1 < \bar{n} < 1$ whereas we need a function to replace the sigmoid used in blocking gates, requiring bounds of $0 < \bar{n} < 1$. This can be achieved using equation 4.

$$F\left(\bar{n}\right) = \min\left(\frac{1}{2\max\left(\left|\bar{n}_1\right|, \left|\bar{n}_2\right|, ..., \left|\bar{n}_i\right|\right)}, 0.5\right) \times \bar{n} + 0.5\bar{j} \tag{4}$$

Where $\bar{j}$ is a vector of ones with the same number of components as $\bar{n}$. This is our hyper-cubic projection equation which meets the projection criteria unlike a typical minimum function. However, this function does not meet spherical symmetry criteria. Equation 3 meets the 'origin at the centre of the output space' criteria but equation 4 does not. To finally create a function analogous to the sigmoid activation for the purpose of a blocking gate, which meets more of the criteria we arrive at equation 5.

$$F(\hat{n}) = C(T(\hat{n})) \tag{5}$$

Where $C(\bar{n})$ is equal $F(\bar{n})$ in equation 5 and $T(\bar{n})$ is equal to a modified Geeb's Tanh Function shown in equation 6. In short $T(\bar{n})$ is equal to $\sqrt{2}$ times equation 1.

$$T(x) = \sqrt{2}\hat{n}\tanh\left|\bar{n}\right| \tag{6}$$

Therefore, we have constructed a function analogous to sigmoid, meeting more criteria than the typical sigmoid. However, this final function still breaks several criteria. Despite this, the criteria are guidance for how to construct generalised activation functions, yet this function is specific purposed and its purpose directly contradicts the criteria it has broken. Therefore it is acceptable as it is optimised to the greatest it can be according to the criteria. Specific purpose

activation functions are always preferable but it is not always feasable to design them.

We have now offered alternatives to the most common activation functions used in fully connected networks, convolutional networks and recurrent neural networks. This final equation means that all activation functions within an standard LSTM now have analogous functions which fulfill our criteria.

3 Discussion

It is evident that many of these functions are substantially more complex than the common activation functions. However, our hope is that testing will show the benefit of using our modified functions far outweighs the partially increased computational processing cost for our new slightly more complex functions. Rudimentary testing by ourselves has indicated that this is true.

We do not believe this list of criteria to be complete; further advancements in understanding the spatial nature of networks will hopefully reveal some further optimisations to activation functions.

Finally, there are three aspects we would like to benchmark our improved functions on in regards to the network's learning capability. The first is the rate of learning whilst using our modified functions compared to common activation functions. Secondly we would like to investigate whether the learning 'saturates' at a better level than a typical activation function. This could be done using accuracy on a classification task. Thirdly, we would expect the time for the network to show initial, measurable improvement on a task to reduce. For example a network will by chance get roughly 10% on the MNIST dataset. The time taken for a statistically significant increase in success above this random chance should be shorter with our new functions.