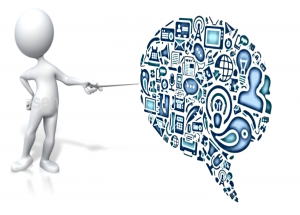
# Fundamentals of Deep Learning – Activation Functions and When to Use Them? 深度学习的基础-激活功能和何时使用它们？

,  
,

OCTOBER 23, 2017  
2017年10月23日

## Introduction 介绍

Internet provides access to plethora of information today. Whatever we need is just a Google (search) away. However, when we have so much of information, the challenge is to segregate between relevant and irrelevant information.  
今天因特网提供了获取大量信息的途径。我们需要的只是谷歌搜索。然而，当我们拥有如此多的信息时，我们面临的挑战是将相关信息和不相关信息分开。



img  
img公司

When our brain is fed with a lot of information simultaneously, it tries hard to understand and classify the information between useful and not-so-useful information. We need a similar mechanism to classify incoming information as useful or less-useful in case of Neural Networks.  
当我们的大脑同时被大量的信息所喂养时，它就很难在有用信息和无用信息之间理解和分类。我们需要一个类似的机制来将输入的信息分类为有用或不太有用的神经网络。

This is a very important in the way a network learns because not all information is equally useful. Some of it is just noise. Well, activation functions help the network do this segregation. They help the network use the useful information and suppress the irrelevant data points.  
这在网络学习中是非常重要的，因为并非所有的信息都同样有用。有些只是噪音。好吧，激活功能有助于网络实现这种隔离。它们有助于网络利用有用的信息，抑制不相关的数据点。

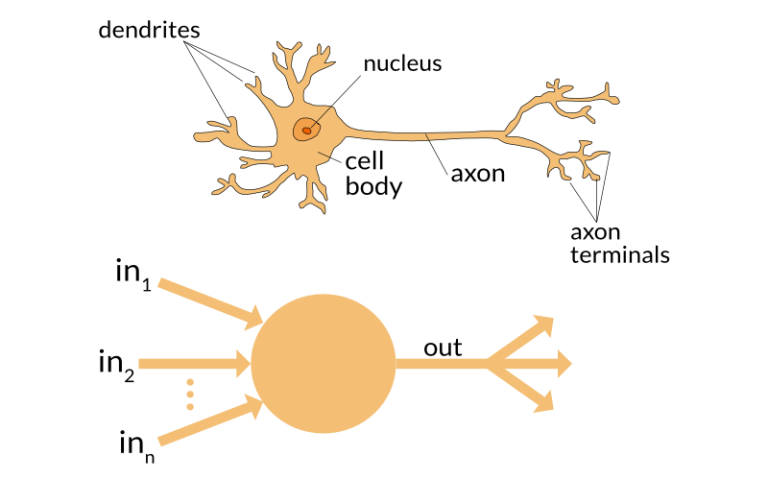
Let us go through these activation functions, how they work and figure out which activation functions fits well into what kind of problem statement.  
让我们来看看这些激活函数，它们是如何工作的，并找出哪些激活函数适合于哪种问题陈述。

## Table of Contents 目录

1. Brief overview of neural networks  
   神经网络概述
2. What is an activation function ?  
   什么是激活功能？
3. Can we do without an activation function ?  
   我们可以不用激活功能吗？
4. Popular types of activation functions and when to use them  
   常用的激活函数类型和何时使用它们
   1. Identity  
      身份
   2. Binary Step  
      二进制步进
   3. Sigmoid  
      乙状结肠
   4. Tanh  
      坦恩
   5. ReLU  
      雷卢
   6. Leaky ReLU  
      漏雷卢
   7. Softmax  
      Softmax公司
5. Choosing the Right Activation Function  
   选择正确的激活功能

## Brief overview of neural networks 神经网络概述

Before I delve into the details of activation functions, let’s do a little review of what are neural networks and how they function. A neural network is a very powerful machine learning mechanism which basically mimics how a human brain learns. The brain receives the stimulus from the outside world, does the processing on the input, and then generates the output.  
在深入研究激活函数的细节之前，让我们先回顾一下什么是神经网络以及它们是如何工作的。神经网络是一种非常强大的机器学习机制，它基本上模仿人脑的学习方式。大脑接收外界的刺激，对输入进行处理，然后产生输出。



img  
img公司

As the task gets complicated multiple neurons form a complex network, passing information among themselves.  
当任务变得复杂时，多个神经元形成一个复杂的网络，在它们之间传递信息。

Using a artificial neural network, we try to mimic a similar behavior. The network you see below is a neural network made of interconnected neurons.  
使用人工神经网络，我们试图模仿类似的行为。你在下面看到的网络是一个由相互连接的神经元组成的神经网络。

img  
img公司

img  
img公司

The black circles in the picture above are neurons. Each neuron is characterized by its weight, bias and activation function. The input is fed to the input layer. The neurons do a linear transformation on the input by the weights and biases. The non linear transformation is done by the activation function. The information moves from the input layer to the hidden layers. The hidden layers would do the processing and send the final output to the output layer. This is the forward movement of information known as the forward propagation. But what if the output generated is far away from the expected value? In a neural network, we would update the weights and biases of the neurons on the basis of the error. This process is known as back-propagation. Once the entire data has gone through this process, the final weights and biases are used for predictions.  
上图中的黑色圆圈是神经元。每个神经元都有其重量、偏倚和激活功能。输入被输入到输入层。神经元通过权值和偏差对输入进行线性变换。非线性变换由激活函数完成。信息从输入层移动到隐藏层。隐藏层将进行处理并将最终输出发送到输出层。这是被称为正向传播的信息的正向运动。但如果产出离预期值很远呢？在神经网络中，我们会根据误差来更新神经元的权值和偏差。这个过程称为反向传播。一旦整个数据经过这个过程，最终的权重和偏差就被用于预测。

## What is an Activation Function? 什么是激活功能？

Activation functions are an extremely important feature of the artificial neural networks. They basically decide whether a neuron should be activated or not. Whether the information that the neuron is receiving is relevant for the given information or should it be ignored.  
激活函数是人工神经网络的一个非常重要的特征。它们基本上决定了一个神经元是否应该被激活。神经元接收到的信息是与给定的信息相关，还是应该被忽略。

img  
img公司

img  
img公司

The activation function is the non linear transformation that we do over the input signal. This transformed output is then sen to the next layer of neurons as input.  
激活函数是我们对输入信号进行的非线性变换。转换后的输出作为输入发送到下一层神经元。

## Can we do without an activation function? 我们可以不用激活功能吗？

Now the question which arises is that if the activation function increases the complexity so much, can we do without an activation function?  
现在出现的问题是，如果激活函数增加了这么多的复杂性，我们能没有激活函数吗？

When we do not have the activation function the weights and bias would simply do a linear transformation. A linear equation is simple to solve but is limited in its capacity to solve complex problems. A neural network without an activation function is essentially just a linear regression model. The activation function does the non-linear transformation to the input making it capable to learn and perform more complex tasks. We would want our neural networks to work on complicated tasks like language translations and image classifications. Linear transformations would never be able to perform such tasks.  
当我们没有激活函数时，权重和偏差只会做一个线性变换。线性方程解起来简单，但求解复杂问题的能力有限。没有激活函数的神经网络本质上只是一个线性回归模型。激活函数对输入进行非线性转换，使其能够学习和执行更复杂的任务。我们希望我们的神经网络能够处理复杂的任务，比如语言翻译和图像分类。线性转换永远无法执行此类任务。

Activation functions make the back-propagation possible since the gradients are supplied along with the error to update the weights and biases. Without the differentiable non linear function, this would not be possible.  
激活函数使反向传播成为可能，因为梯度和误差一起提供，以更新权重和偏差。没有可微的非线性函数，这是不可能的。

## Popular types of activation functions and when to use them 常用的激活函数类型和何时使用它们

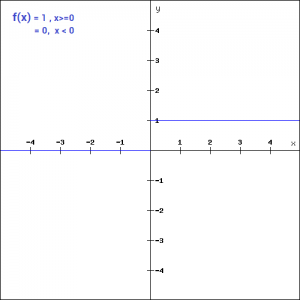
### Binary Step Function 二元阶跃函数

The first thing that comes to our mind when we have an activation function would be a threshold based classifier i.e. whether or not the neuron should be activated. If the value Y is above a given threshold value then activate the neuron else leave it deactivated.  
当我们有一个激活函数时，首先想到的是一个基于阈值的分类器，即神经元是否应该被激活。如果Y值高于给定的阈值，则激活神经元，否则保持其不激活。

It is defined as –  
定义为-

f(x) = 1, x>=0  
f（x）=1，x>=0

= 0, x<0  
=0，x<0



img  
img公司

The binary function is extremely simple. It can be used while creating a binary classifier. When we simply need to say yes or no for a single class, step function would be the best choice, as it would either activate the neuron or leave it to zero.  
二进制函数非常简单。它可以在创建二进制分类器时使用。当我们只需要对一个类说“是”或“否”时，step函数将是最佳选择，因为它要么激活神经元，要么将其设为零。

The function is more theoretical than practical since in most cases we would be classifying the data into multiple classes than just a single class. The step function would not be able to do that.  
该函数理论性强，实用性强，因为在大多数情况下，我们会将数据分类为多个类，而不仅仅是一个类。步骤函数将无法执行此操作。

Moreover, the gradient of the step function is zero. This makes the step function not so useful since during back-propagation when the gradients of the activation functions are sent for error calculations to improve and optimize the results. The gradient of the step function reduces it all to zero and improvement of the models doesn’t really happen.  
此外，阶跃函数的梯度为零。这使得阶跃函数不那么有用，因为在反向传播过程中，当激活函数的梯度被发送用于错误计算以改进和优化结果时阶跃函数的梯度将其降为零，模型的改进并没有真正实现。

f '(x) = 0, for all x  
f’（x）=0，对于所有x

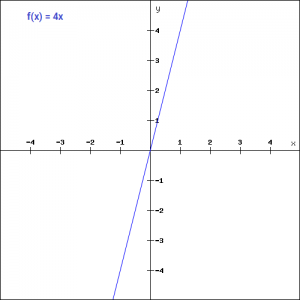
img  
img公司

img  
img公司

### Linear Function 线性函数

We saw the problem with the step function, the gradient being zero, it was impossible to update gradient during the backpropagation. Instead of a simple step function, we can try using a linear function. We can define the function as-  
我们看到了阶跃函数的问题，梯度为零，在反向传播过程中不可能更新梯度。我们可以尝试使用线性函数，而不是简单的阶跃函数。我们可以将函数定义为-

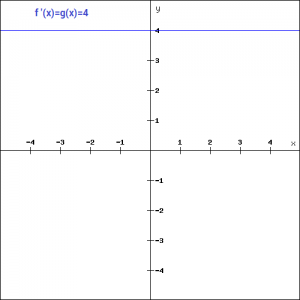
f(x)=ax  
f（x）=ax



img  
img公司

We have taken a as 4 in the figure above. Here the activation is proportional to the input. The input x, will be transformed to ax. This can be applied to various neurons and multiple neurons can be activated at the same time. Now, when we have multiple classes, we can choose the one which has the maximum value. But we still have an issue here. Let’s look at the derivative of this function.  
我们把上图中的a取为4。这里的激活与输入成正比。输入x，将转换为ax。这可以应用于不同的神经元，同时可以激活多个神经元。现在，当我们有多个类时，我们可以选择一个具有最大值的类。但我们仍然有一个问题。让我们看看这个函数的导数。

f'(x) = a  
f’（x）=a



img  
img公司

The derivative of a linear function is constant i.e. it does not depend upon the input value x.  
线性函数的导数是常数，即它不依赖于输入值x。

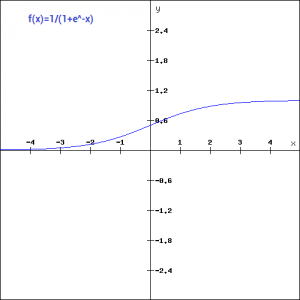
This means that every time we do a back propagation, the gradient would be the same. And this is a big problem, we are not really improving the error since the gradient is pretty much the same. And not just that suppose we are trying to perform a complicated task for which we need multiple layers in our network. Now if each layer has a linear transformation, no matter how many layers we have the final output is nothing but a linear transformation of the input. Hence, linear function might be ideal for simple tasks where interpretability is highly desired.  
这意味着每次我们做反向传播，梯度都是一样的。这是一个大问题，我们并没有真正改善误差，因为梯度几乎相同。不仅仅是假设我们要执行一个复杂的任务，我们需要在我们的网络中有多个层。现在如果每一层都有一个线性变换，不管我们有多少层，最终的输出只是输入的一个线性变换。因此，线性函数可能是非常需要可解释性的简单任务的理想选择。

### Sigmoid 乙状结肠

Sigmoid is a widely used activation function. It is of the form-  
乙状结肠是一种广泛应用的激活功能。它的形式-

f(x)=1/(1+e^-x)  
f（x）=1/（1+e^-x）

Let’s plot this function and take a look of it.  
让我们来绘制这个函数并查看它。

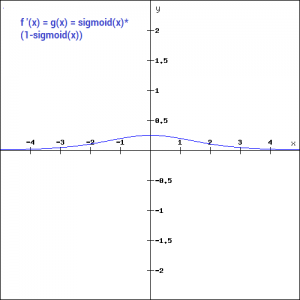


img  
img公司

This is a smooth function and is continuously differentiable. The biggest advantage that it has over step and linear function is that it is non-linear. This is an incredibly cool feature of the sigmoid function. This essentially means that when I have multiple neurons having sigmoid function as their activation function – the output is non linear as well. The function ranges from 0-1 having an S shape. Let’s take a look at the shape of the curve. The gradient is very high between the values of -3 and 3 but gets much flatter in other regions. How is this of any use?  
这是一个光滑函数，是连续可微的。它相对于阶跃函数和线性函数最大的优点是它是非线性的。这是乙状结肠功能的一个非常酷的特征。这基本上意味着当我有多个具有乙状结肠功能的神经元作为它们的激活功能时，输出也是非线性的。函数的范围为0-1，S形。让我们看看曲线的形状。在-3和3之间的梯度非常高，但在其他区域会变得更平坦。这有什么用？

This means that in this range small changes in x would also bring about large changes in the value of Y. So the function essentially tries to push the Y values towards the extremes. This is a very desirable quality when we’re trying to classify the values to a particular class.  
这意味着在这个范围内，x的微小变化也会导致Y值的大变化，因此函数本质上试图将Y值推向极端。当我们试图将值分类到一个特定的类时，这是一个非常理想的质量。

Let’s take a look at the gradient of the sigmoid function as well.  
让我们来看看乙状结肠的梯度函数。



img  
img公司

It’s smooth and is dependent on x. This means that during backpropagation we can easily use this function. The error can be backpropagated and the weights can be accordingly updated.  
它是光滑的并且依赖于x。这意味着在反向传播过程中我们可以很容易地使用这个函数。误差可以反向传播，权重也可以相应地更新。

Sigmoids are widely used even today but we still have a problems that we need to address. As we saw previously – the function is pretty flat beyond the +3 and -3 region. This means that once the function falls in that region the gradients become very small. This means that the gradient is approaching to zero and the network is not really learning.  
乙状结肠即使在今天也被广泛使用，但我们仍有一个问题需要解决。正如我们之前看到的，函数在+3和-3区域之外是非常平坦的。这意味着一旦函数落在该区域，梯度就变得非常小。这意味着梯度接近于零，网络并不是真正的学习。

Another problem that the sigmoid function suffers is that the values only range from 0 to 1. This means that the sigmoid function is not symmetric around the origin and the values received are all positive. So not all times would we desire the values going to the next neuron to be all of the same sign. This can be addressed by scaling the sigmoid function. That’s exactly what happens in the tanh function. let’s read on.  
sigmoid函数遇到的另一个问题是，值的范围仅为0到1。这意味着乙状结肠功能在原点周围不对称，并且接收到的值都是正的。所以不是所有时候我们都希望下一个神经元的值是相同的。这可以通过缩放乙状结肠功能来解决。这正是tanh函数中发生的事情。让我们继续读下去。

### Tanh 坦恩

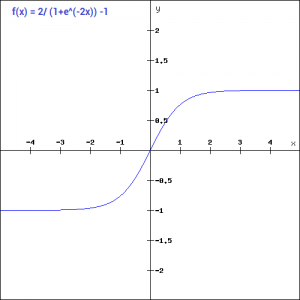
The tanh function is very similar to the sigmoid function. It is actually just a scaled version of the sigmoid function.  
tanh函数与乙状结肠非常相似。它实际上只是乙状结肠功能的一个缩放版本。

tanh(x)=2sigmoid(2x)-1  
tanh（x）=2乙状体（2x）-1

It can be directly written as –  
它可以直接写成-

tanh(x)=2/(1+e^(-2x)) -1  
tanh（x）=2/（1+e^（-2x））-1

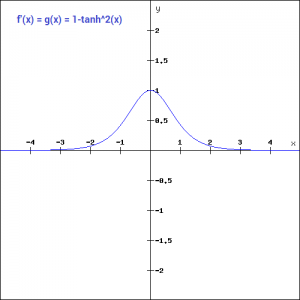
Tanh works similar to the sigmoid function but is symmetric over the origin. it ranges from -1 to 1.  
Tanh的工作原理与sigmoid函数相似，但在原点上是对称的。范围从-1到1。



img  
img公司

It basically solves our problem of the values all being of the same sign. All other properties are the same as that of the sigmoid function. It is continuous and differentiable at all points. The function as you can see is non linear so we can easily backpropagate the errors.  
它基本上解决了我们的价值观都是同一个符号的问题。所有其他性质与乙状结肠的性质相同。它在所有点上都是连续可微的。如你所见的函数是非线性的，所以我们可以很容易地反向传播错误。

Let’s have a look at the gradient of the tan h function.  
让我们看看tan h函数的梯度。



img  
img公司

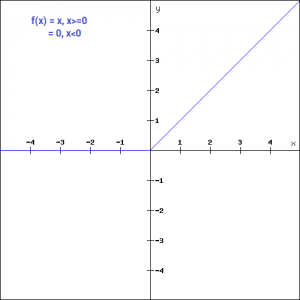
The gradient of the tanh function is steeper as compared to the sigmoid function. Our choice of using sigmoid or tanh would basically depend on the requirement of gradient in the problem statement. But similar to the sigmoid function we still have the vanishing gradient problem. The graph of the tanh function is flat and the gradients are very low.  
tanh函数的梯度比sigmoid函数更陡。我们选择使用乙状结肠还是tanh基本上取决于问题陈述中对梯度的要求。但与乙状窦函数类似，我们仍然有消失梯度问题。tanh函数的图形是平坦的，梯度很低。

### ReLU 雷卢

The ReLU function is the Rectified linear unit. It is the most widely used activation function. It is defined as-  
ReLU函数是经过校正的线性单元。它是应用最广泛的激活函数。它被定义为-

f(x)=max(0,x)  
f（x）=最大值（0，x）

It can be graphically represented as-  
它可以用图形表示为-

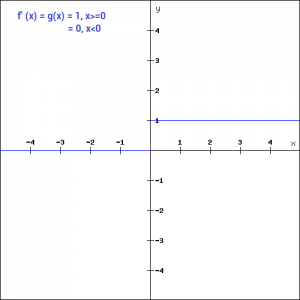


img  
img公司

ReLU is the most widely used activation function while designing networks today. First things first, the ReLU function is non linear, which means we can easily backpropagate the errors and have multiple layers of neurons being activated by the ReLU function.  
ReLU是当今网络设计中应用最广泛的激活函数。首先，ReLU函数是非线性的，这意味着我们可以很容易地反向传播错误，并且有多层神经元被ReLU函数激活。

The main advantage of using the ReLU function over other activation functions is that it does not activate all the neurons at the same time. What does this mean ? If you look at the ReLU function if the input is negative it will convert it to zero and the neuron does not get activated. This means that at a time only a few neurons are activated making the network sparse making it efficient and easy for computation.  
与其他激活函数相比，使用ReLU函数的主要优点是它不会同时激活所有的神经元。这是什么意思？如果你看ReLU函数，如果输入是负的，它将把它转换为零，神经元不会被激活。这意味着一次只有少数神经元被激活，使得网络稀疏，从而使其高效且易于计算。

Let’s look at the gradient of the ReLU function.  
让我们看看ReLU函数的梯度。



img  
img公司

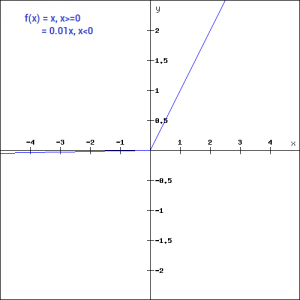
But ReLU also falls a prey to the gradients moving towards zero. If you look at the negative side of the graph, the gradient is zero, which means for activations in that region, the gradient is zero and the weights are not updated during back propagation. This can create dead neurons which never get activated. When we have a problem, we can always engineer a solution.  
但雷卢也成为向零移动的梯度的牺牲品。如果查看图的负边，渐变为零，这意味着对于该区域中的激活，渐变为零，并且权重在反向传播期间不会更新。这会产生永远不会被激活的死亡神经元。当我们遇到问题时，我们总能想出解决办法。

### Leaky ReLU 漏雷卢

Leaky ReLU function is nothing but an improved version of the ReLU function. As we saw that for the ReLU function, the gradient is 0 for x<0, which made the neurons die for activations in that region. Leaky ReLU is defined to address this problem. Instead of defining the Relu function as 0 for x less than 0, we define it as a small linear component of x. It can be defined as-  
Leaky ReLU函数只是ReLU函数的一个改进版本。正如我们所看到的，对于ReLU函数，x<0的梯度为0，这使得该区域的神经元因激活而死亡。Leaky ReLU被定义为解决这个问题。对于小于0的x，Relu函数不是定义为0，而是定义为x的一个小线性分量-

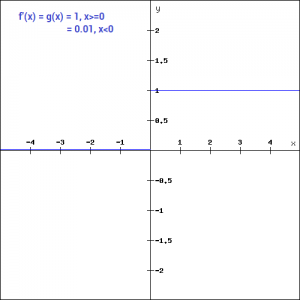
f(x)= ax, x<0  
= x, x>=0  
f（x）=ax，x<0=x，x>=0

What we have done here is that we have simply replaced the horizontal line with a non-zero, non-horizontal line. Here a is a small value like 0.01 or so. It can be represented on the graph as-  
我们在这里所做的就是用一条非零的非水平线来代替水平线。这里a是一个很小的值，比如0.01左右。它可以在图上表示为-



img  
img公司

The main advantage of replacing the horizontal line is to remove the zero gradient. So in this case the gradient of the left side of the graph is non zero and so we would no longer encounter dead neurons in that region. The gradient of the graph would look like –  
替换水平线的主要优点是消除零梯度。在这种情况下，图左侧的梯度是非零的，因此我们将不再在该区域遇到死亡的神经元。图的梯度看起来像-



img  
img公司

Similar to the Leaky ReLU function, we also have the Parameterised ReLU function. It is defined similar to the Leaky ReLU as –  
与Leaky ReLU函数类似，我们也有参数化的ReLU函数。其定义类似于泄漏ReLU，即-

f(x)= ax, x<0  
= x, x>=0  
f（x）=ax，x<0=x，x>=0

However, in case of a parameterised ReLU function, ‘a‘ is also a trainable parameter. The network also learns the value of ‘a‘ for faster and more optimum convergence. The parametrised ReLU function is used when the leaky ReLU function still fails to solve the problem of dead neurons and the relevant information is not successfully passed to the next layer.  
但是，对于参数化的ReLU函数，“a”也是一个可训练的参数。该网络还学习了“a”的值，以便更快、更优地收敛。当泄漏的ReLU函数仍然不能解决神经元死亡的问题，且相关信息未能成功传递到下一层时，使用参数化ReLU函数。

### Softmax Softmax公司

The softmax function is also a type of sigmoid function but is handy when we are trying to handle classification problems. The sigmoid function as we saw earlier was able to handle just two classes. What shall we do when we are trying to handle multiple classes. Just classifying yes or no for a single class would not help then. The softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs. This essentially gives the probability of the input being in a particular class. It can be defined as –  
softmax函数也是一种sigmoid函数，但在处理分类问题时非常方便。我们前面看到的sigmoid函数只能处理两个类。当我们试图处理多个类时，我们该怎么办。仅仅对一个类的是或否进行分类是没有帮助的。softmax函数将每个类的输出压缩到0到1之间，还将除以输出的和。这基本上给出了输入在特定类中的概率。它可以定义为-

img

img  
img公司

Let’s say for example we have the outputs as-  
例如，我们将输出作为-

[1.2 , 0.9 , 0.75], When we apply the softmax function we would get [0.42 , 0.31, 0.27]. So now we can use these as probabilities for the value to be in each class.  
[1.2，0.9，0.75]，当我们应用softmax函数时，我们将得到[0.42，0.31，0.27]。所以现在我们可以用这些作为每个类中的值的概率。

The softmax function is ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.  
softmax函数理想地用于分类器的输出层，在这里我们实际上试图获得定义每个输入的类的概率。

## Choosing the right Activation Function 选择正确的激活功能

Now that we have seen so many activation functions, we need some logic / heuristics to know which activation function should be used in which situation. Good or bad – there is no rule of thumb.  
既然我们已经看到了这么多的激活函数，我们需要一些逻辑/启发式方法来知道在什么情况下应该使用哪个激活函数。好的或坏的-没有经验法则。

However depending upon the properties of the problem we might be able to make a better choice for easy and quicker convergence of the network.  
但是，根据问题的性质，我们可以做出更好的选择，使网络更容易和更快地收敛。

* Sigmoid functions and their combinations generally work better in the case of classifiers  
  对于分类器来说，乙状窦函数及其组合通常效果更好
* Sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem  
  由于消失梯度问题，有时可以避免使用乙状结肠和tanh函数
* ReLU function is a general activation function and is used in most cases these days  
  ReLU函数是一个通用的激活函数，目前在大多数情况下使用
* If we encounter a case of dead neurons in our networks the leaky ReLU function is the best choice  
  如果我们在我们的网络中遇到一个神经元死亡的情况，泄漏的ReLU函数是最好的选择
* Always keep in mind that ReLU function should only be used in the hidden layers  
  请记住，ReLU函数只能在隐藏层中使用
* As a rule of thumb, you can begin with using ReLU function and then move over to other activation functions in case ReLU doesn’t provide with optimum results  
  根据经验，您可以从使用ReLU函数开始，然后转到其他激活函数，以防ReLU不能提供最佳结果

## Projects 项目

Now, its time to take the plunge and actually play with some other real datasets. So are you ready to take on the challenge? Accelerate your deep learning journey with the following Practice Problems:  
现在，是时候冒险去玩一些其他真实的数据集了。你准备好接受挑战了吗？通过以下实践问题加快您的深度学习之旅：

|  |  |  |
| --- | --- | --- |
| [img](https://datahack.analyticsvidhya.com/contest/practice-problem-identify-the-apparels/?utm_source=fundamentals-deep-learning-activation-functions-when-to-use-them&utm_medium=blog) | [Practice Problem: Identify the Apparels](https://datahack.analyticsvidhya.com/contest/practice-problem-identify-the-apparels/?utm_source=fundamentals-deep-learning-activation-functions-when-to-use-them&utm_medium=blog) | Identify the type of apparel for given images 识别给定图像的服装类型 |
| [img](https://datahack.analyticsvidhya.com/contest/practice-problem-identify-the-digits/?utm_source=fundamentals-deep-learning-activation-functions-when-to-use-them&utm_medium=blog) | [Practice Problem: Identify the Digits](https://datahack.analyticsvidhya.com/contest/practice-problem-identify-the-digits/?utm_source=fundamentals-deep-learning-activation-functions-when-to-use-them&utm_medium=blog) | Identify the digit in given images |

## End Notes 尾注

In this article I have discussed the various types of activation functions and what are the types of problems one might encounter while using each of them.  
在本文中，我讨论了各种类型的激活函数，以及在使用它们时可能遇到的问题类型。

I would suggest to begin with a ReLU function and explore other functions as you move further. You can also design your own activation functions giving a non-linearity component to your network. If you have used your own activation function which worked really well, please share it with us and we shall be happy to incorporate it into the list.  
我建议从ReLU函数开始，随着您的进一步深入，探索其他函数。您还可以设计自己的激活函数，为您的网络提供非线性组件。如果你已经使用了你自己的激活功能，这是非常好的工作，请与我们分享，我们将很高兴把它纳入名单。