计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(7):43−49 [doi: 10.15888/j.cnki.csa.006463]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

激活函数在卷积神经网络中的对比研究①

田　娟, 李英祥, 李彤岩

(成都信息工程大学 通信工程学院, 成都 610225)
通讯作者: 田　娟, E-mail: 1849336438@qq.com

摘　要: 近年, 深度学习的快速发展致使越来越多的人从事相关的研究工作. 但是, 许多研究者在搭建深度神经网络模型时只是根据标准算法或改进算法直接搭建, 而对算法本身及影响模型性能的因素不甚了解, 致使在许多应用中或多或少存在盲目套用现象. 通过研究深度神经网络, 选择其中的重要影响因素激活函数进行深入研究. 首先, 分析了激活函数如何影响深度神经网络; 接着对激活函数的发展现状及不同激活函数的原理性能进行了分析总结; 最后, 基于 Caffe 框架用 CNN 对 Mnist 数据集进行分类识别实验, 对 5 种常用激活函数进行综合分析比较, 为设计深度神经网络模型时选用激活函数提供参考.

关键词: 卷积神经网络; 激活函数; Caffe; 梯度下降法; 网络性能

引用格式: 田娟,李英祥,李彤岩.激活函数在卷积神经网络中的对比研究.计算机系统应用,2018,27(7):43–49. http://www.c-s-a.org.cn/1003-3254/6463.html

Contrastive Study of Activation Function in Convolutional Neural Network

TIAN Juan, LI Ying-Xiang, LI Tong-Yan

(School of Communication Engineering, Chengdu University of Information Technology, Chengdu 610225, China)

Abstract: In recent years, the rapid development of deep learning has led more and more people to engage in related research work. However, many researchers construct deep neural network models based on standard algorithms or improved algorithms, but do not understand the algorithm itself and the factors that affect the performance of the model, resulting in more or less blind application in many applications. By studying the deep neural network, the activation function of the important influencing factors was studied. First, the activation function is analyzed to influence the depth neural network. Then, the development of activation function and the principle and performance of different activation functions are analyzed and summarized. Finally, based on the Caffe framework, the CNN is used to classify and identify MNIST data sets. Five kinds of commonly used activation functions are analyzed and compared comprehensively to provide a reference for the selection of activation function in the design of deep neural network model.

Key words: convolutional neural network; activation function; Caffe; gradient method; network performance

1　引言

近年, 深度学习[1]在计算机视觉、语音识别、自然语言处理、图形图像识别[2]等领域的运用备受关注. 与浅层神经网络相比, 深度神经网络层数的增加意味着其学到的特征越紧密, 表达能力也越强. 但是, 训练的层数越多, 训练也就越困难, 主要是因为在反向传播算法中残差会随着网络传播的深度递减, 使得底层网络因为残差过小而无法得到有效训练或无法训练. 残差的衰减与网络模型中激活函数 (activation function) 的选用密切相关, 更好的激活函数可以抑制残差在网络

① 基金项目: 四川省科技厅重点研发项目 (2017FZ0100)
Foundation item: Key Development Program of the Ministry of Science and Technology of Sichuan Province (2017FZ0100)
收稿时间: 2017-10-26; 修改时间: 2017-11-14; 采用时间: 2018-01-08; csa 在线出版时间: 2018-05-24

传播过程中的衰减并提高模型的收敛速度. 深度神经网络之所以能得到快速发展和应用很大一部分得益于激活函数的发展. 文章针对激活函数进行了深入研究, 对激活函数如何影响深度网络的训练, 其发展现状及原理性能进行了分析总结, 并基于 Caffe 框架用 CNN 对 Mnist 数据集进行分类识别实验. 在整个实验过程中, 激活函数是唯一的变量. 通过实验验证不同激活函数对深度神经网络带来的影响.

2 激活函数如何影响神经网络

神经网络中激活函数的主要作用是提供网络的非线性建模能力. 如果网络中仅包含线性卷积和全连接运算, 网络仅能够表达线性映射, 即便增加网络的深度也依旧是线性映射, 难以有效建模实际环境中非线性分布的数据. 当激活函数应用于深度神经网络时, 其主要在前向传播和反向传播两个过程中对网络训练产生影响.

图 1 是神经网络的基本处理单元.

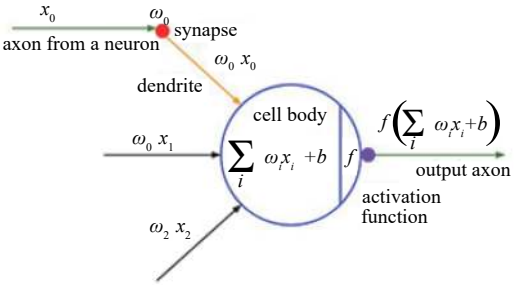

图 1 神经元前向传播示意图

图 1 的输出为:

$$output = f(\Sigma\omega_i x_i + b) \tag{1}$$

其中, ω_i是任意初始的权值向量, x_i是输入向量, b 是该神经元的偏置值, $f(x)$是激活函数. *output* 在神经网络中为下一个神经元的输入. 从式 (1) 可以看出, 输入向量首先与权值向量做内积, 内积和再加上偏置项, 最后经过一个激活函数$f(x)$作用再输出. 激活函数在这里的作用是把输入数据的特征通过自身的非线性特性保留并映射出来, 传入下一个神经元. 在前向传播中, 样本从输入层到隐藏层再到输出层的整个数据传输过程都是激活函数利用其信息处理特性将数据进行变换衔接起来的.

反向传播过程是网络更新权值和偏置值的过程. 图 2 是一个输出神经元的反向传播过程.

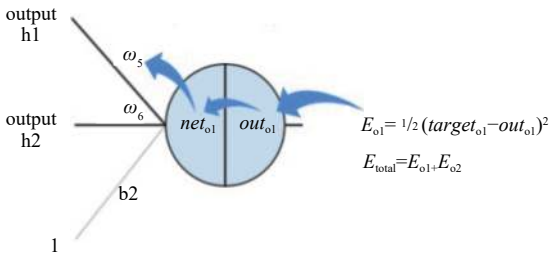

图 2 神经元反向传播示意图

假设整个网络的输出层有两个神经元, 则网络输出误差表示为:

$$\begin{aligned} E_{\text{total}} &= E_{\text{o1}} + E_{\text{o2}} \\ &= \frac{1}{2}\left[(target_{\text{o1}} - out_{\text{o1}})^2 + (target_{\text{o2}} - out_{\text{o2}})^2\right] \end{aligned} \tag{2}$$

反向传播过程通过梯度下降算法对ω_5进行调整时需要求$\frac{\partial E_{\text{total}}}{\partial \omega_5}$.

通过链式法则有:

$$\frac{\partial E_{\text{total}}}{\partial \omega_5} = \frac{\partial E_{\text{total}}}{\partial out_{\text{o1}}}\frac{\partial out_{\text{o1}}}{\partial net_{\text{o1}}}\frac{\partial net_{\text{o1}}}{\partial \omega_5} \tag{3}$$

其中,

$$\begin{aligned} &\frac{\partial E_{\text{total}}}{\partial out_{\text{o1}}} \\ &= \frac{\partial}{\partial out_{\text{o1}}}\left[\frac{1}{2}(target_{\text{o1}} - out_{\text{o1}})^2 + \frac{1}{2}(target_{\text{o2}} - out_{\text{o2}})^2\right] \end{aligned} \tag{4}$$

$$\frac{\partial out_{\text{o1}}}{\partial net_{\text{o1}}} = f'(net_{\text{o1}}) \tag{5}$$

$$\frac{\partial net_{\text{o1}}}{\partial \omega_5} = \frac{\partial}{\partial \omega_5}(\omega_5 \times out_{\text{h1}} + \omega_6 \times out_{\text{h2}} + b_2 \times 1) \tag{6}$$

则ω_5的更新方程为:

$$\omega_5' = \omega_5 - \eta\frac{\partial E_{\text{total}}}{\partial \omega_5} \tag{7}$$

网络中其他的权值更新和上述过程类似. 从式 (5) 可以看出, 在每一次更新中, 每经过一层, 残差都要乘以$f'(x)$. 如果选择的激活函数不恰当, 经过多层以后深度神经网络就会因底层残差太小而无法得到有效训练. 因此, 在反向传播过程中, 激活函数的选择会影响到整个模型是否收敛和收敛效果.

3 激活函数的发展及现状

激活函数的前期研究经历了三个阶段:

(1) 简单线性模型: 这种模型反映不出激活函数的非线性特征, 不具备分类的特性, 因此很少被采用.

(2) 线性阈值函数: 该函数具有良好的分类特性, 但由于其为不可导函数, 使得难以找到有效的学习算法.

(3) Sigmoid 函数

Sigmoid 函数是典型的 S 型函数, 函数表达式为:

$$\varphi(x)=\frac{1}{1+e^{-x}} \tag{8}$$

函数图像如图 3 所示.

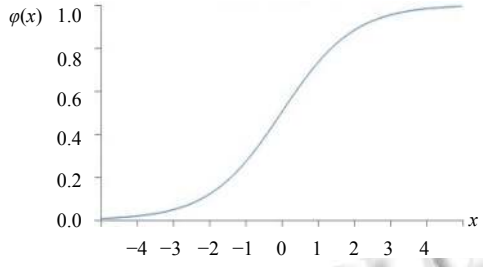

图 3 Sigmoid 函数图像

从图 3 可以看出, Sigmoid 函数的输出值在 (0,1) 之间, 为非线性函数. 当 x 的取值越来越大时, 函数曲线变得越来越平缓, 意味着此时函数导数$\phi'(x)$也越来越小. 同样的, 当 x 的取值越来越小时也存在这样的问题. 只有当 x 取值在 0 附近时, 导数$\phi'(x)$取值较大. Sigmoid 函数对中间区域的信号增益较大, 两侧区域信号增益较小, 克服了前两种函数的缺陷, 因此在一段时间内得到了广泛地应用. 但是, 在使用过程中其缺点逐渐暴露, 主要表现在两点. (1) 在反向传递过程中, 当输入非常大或者非常小时, $\phi'(x)$趋近于 0, 导致向下一层传递的梯度变得非常小, 从而导致梯度消失. (2) Sigmoid 函数不是关于原点中心对称的, 这个特性会导致后面网络层的输入也不是以零为中心的, 进而影响梯度下降的运作.

针对 Sigmoid 的问题, 后续出现了大量的改进函数. 有使用正弦函数、反正切函数来代替 Sigmoid 函数的, 其中应用最广的是使用双曲正切函数 (Tanh) 来代替 Sigmoid. Tanh 的函数表达式为:

$$\varphi(x)=\frac{1-e^{-2x}}{1+e^{-2x}} \tag{9}$$

函数图像如图 4 所示.

从图 4 可以看出, Tanh 与 Sigmoid 函数图像相似, 都是连续可微的非线性函数, 不同的是 Tanh 的输出以 0 为中心, 函数位于 (−1,1) 区间上. 文献[3]提到 Tanh 网络的收敛速度比 Sigmoid 快. 当 x 非常大或者非常小时, Tanh 函数的倒数仍然趋近于 0, 因此它并没有解决 Sigmoid 函数的最大问题——由于函数饱和性产生的梯度消失.

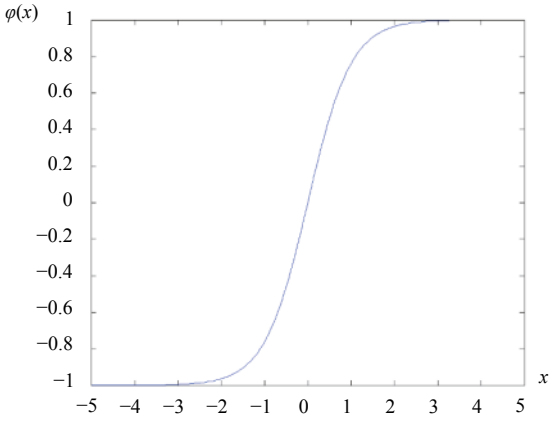

图 4 Tanh 函数图像

Sigmoid、Tanh 激活函数是传统神经网络模型中十分典型, 运用广泛的两个激活函数, 它们的广泛运用形成了一种概念: 非线性激活函数要比线性激活函数更加先进. 然而, 这一观念随着神经元的稀疏性被证实而打破. 神经元对输入的大量信号进行了抑制, 只有少部分信号被响应, 这样可以提高学习的精度, 更好地提取出稀疏特征, 同时稀疏特征并不需要网络具有很强大的线性不可分机制, 因此使用简单的线性激活函数可能更为合适. 另一方面, 2001 年 Charles Dugas 等人在做正数回归预测[4]中偶然使用了 Softplus 函数取得了很好的效果, 同年 Charles Dugas 等人在 NIPS 会议中提到, Softplus 可以看作是强制非负校正函数 max(0, x) 的平滑版本. 这些都为 ReLU 函数的发展奠定了基础. Softplus 和 ReLU 的函数图像如图 5 所示.

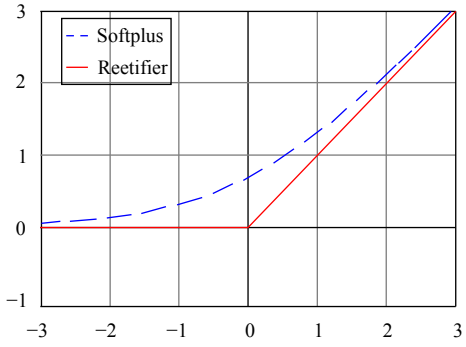

图 5 Softplus 和 ReLU 函数图像

线性激活函数 ReLU[5]在深度神经网络中的使用

加速了深度网络的直接监督式训练的最终突破, 也是 2012 年卷积神经网络在 ILSVRC 竞赛中取得里程碑突破的原因之一. 其在网络中的使用效果明显优于前面的 Sigmoid、Tanh 函数, 是现在使用最广泛的激活函数. 其函数表达式为:

$$f(x)=\max(0,x) \tag{10}$$

由图 5 可知, ReLU 能够在 $x>0$ 时保持梯度不衰减, 从而有效的缓解了梯度消失问题, 而且其运算简单. 另外, ReLU 函数优于其他激活函数的另一优点是它不会同时激活所有的神经元. 如果输入值为负, ReLU 函数会转换为 0, 此时神经元不被激活. 这意味着, 在一段时间内, 只有少量的神经元被激活, 神经网络的这种稀疏性使其变得高效且易于计算. ReLU 在给神经元网络带来稀疏性的同时也带来了弊端. 当输入落入 $x<0$ 的区域, 会导致对应的权值无法更新, 而且强制的稀疏处理也会减少模型的有效容量, 即特征屏蔽太多, 导致模型无法学习到有效特征. 因此 ReLU 在训练过程中非常的脆弱.

为了缓解 ReLU 的问题,随后提出了 LReLU、PReLU、RReLU 等 ReLU 的变形函数. 其中的基本思想是在 $x<0$ 时乘以一个比较小的系数α, 函数表达式为:

$$f(x)=\max(\alpha x,x) \tag{11}$$

函数图像如图 6 所示.

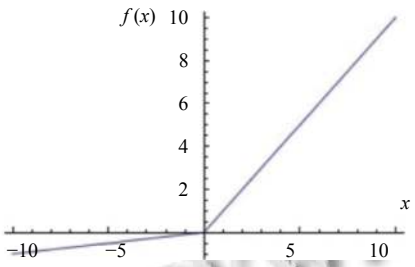

图 6 LReLU/PReLU/RReLU 函数图像

Leaky ReLU(LReLU) 中的α是一个很小的常数, 通常是通过先验知识人工赋值. 相比于 ReLU, Leaky ReLU 在 x 的负半轴引入了参数α, 使得函数在 $x<0$ 区域内倒数不为 0, 这样既修正了数据分布, 又保留了一些负轴的值, 使得负轴信息不会全部丢失. 目前, 关于 Leaky ReLU 的效果众说纷纭, 没有清晰的定论.

2015 年 Kaiming He 在文献[6]中指出, Leaky ReLU 函数中的参数α不仅可以作为超参数进行训练, 而且可以达到更好的效果,于是 Parametric ReLU(PReLU) 函数被提出. PReLU 中α是一个随机的超参数, 其在训练过程中可以进行反向传播从而被学习, 并且在测试时为固定值. 这使神经元能够选择负区域最好的梯度, 有了这种能力, 它可以变成 ReLU 或 Leaky ReLU. 在文献[6]中指出 PReLU 比 ReLU 收敛速度更快. 因为 PReLU 的输出更接近 0 均值, 使得 SGD 更接近 natural gradient.

在 Kaggle 的 NDSB 比赛中 Randomized Leaky ReLU (RReLU)[7]函数被首次提出, 它是 LReLU 的“随机”版本, 即指α是随机的. 其核心思想是在训练过程中α从一个高斯分布$U(l,u)$中随机生成, 然后在测试过程中进行修正.

2013 年, Maxout 激活函数在 ICML2013 上被提出, 作者 Goodfellow 在文献[8]中将 Maxout 与 Dropout 结合, 证明其在 MNIST, CIFAR-10, CIFAR-100, SVHN 这 4 个数据上都取得了很好的识别率. Maxout 是 ReLU 的推广, 其隐含层节点的输出表达式为:

$$f_i(x)=\max_{j\in[1,k]} z_{ij} \tag{12}$$

其中, $z_{ij}=x^{\mathrm{T}}W_{\cdots ij}+b_{ij}, W\in \mathrm{R}^{d\times m\times k}$.

Maxout 引入了参数 k, 与 Sigmoid、ReLU 相比其增加了 k 个神经元, 然后输出激活函数中的最大值. Maxout 是一个可学习的分段线性函数, 具有非常强的拟合能力, 研究表明两层 Maxout 就可以拟合任意的的凸函数. 如 ReLU 和 Leaky ReLU 均可用 Maxout 归纳. 然而与 ReLU 相比, Maxout 每个神经元的参数数量增加了一倍, 这就导致整体参数的数量剧增, 使得网络的计算量增加了.

2015 年 Clevert 等人在文献[9]中提出了 ELU 激活函数, 它是一种融合了 Sigmoid 和 ReLU 的一类激活函数. 函数表达式为:

$$f(x)=\begin{cases}x, & x<0\\ a(\exp(x)-1), & x\geq 0\end{cases} \tag{13}$$

函数图像如图 7 所示.

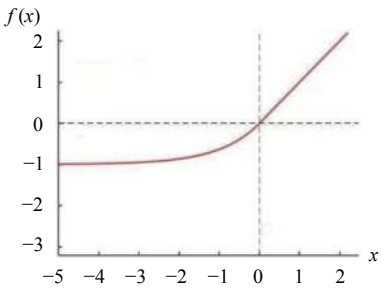

图 7 ELU 函数图像

由图 7 可知, ELU 左侧部分解决了 ReLU 落入“死亡区”的问题, 并且使得 ELU 对输入变化或噪声更具鲁棒性. 右侧线性部分使得 ELU 能够解决 Sigmoid 出现的梯度消失问题.

2017 年 10 月, 在文献[10]中谷歌研究者发布了新型激活函数 Swish, 引发了不小的争议. 其数学公式为:

$$f(x) = x \cdot sigmoid(\beta x) = \frac{x}{1+e^{-\beta x}} \tag{14}$$

其中,β是一个常数或可训练的参数. 函数图像如图 8 所示.

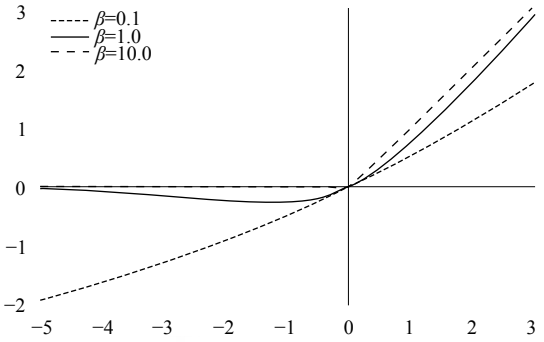

图 8 Swish 函数图像

由图 8 可以看出, Swish 具有无上界有下界、平滑、非单调等特性. 当$\beta=0$时, Swish 为线性函数 $f(x)=x/2$, 当β趋近于无穷时, Swish 为线性函数变为了 ReLU 函数 因此 Swish 可以看成是介于线性函数和 ReLU 之间的平滑函数. 在文献[10]中证明了 Swish 函数的性能优于 ReLU, 但也有研究中认为 Swish 函数并没有什么创新意义. 从函数本身来看在 $x>0$ 区域其曲线形状与 ReLU 激活函数不同, 即使在输入值增大的情况下, Swish 激活函数的输出可能下降. 大多数激活函数是单调的, 即输入值增大的情况下, 输出值不可能下降. 而 Swish 函数为 0 时具备单侧有界的特性. Swish 能否代替标准的 ReLU 还需要进一步研究.

此外, 在文献[11]中提出的 CReLU 激活函数, 它来自 ICML2016, 是对 ReLU 的一种改进激活函数. 其主要思想是将激活函数的输入做一次额外的取反以达到消除冗余滤波器的目的. 文献[12]中提出了另一种激活函数 MPELU, 它同时具备 ReLU、PReLU 和 ELU 的优点. 文献[13]针对 S 型函数会导致学习速度下降问题, 提出了一种组合激活函数, 该函数既满足了 S 型函数的要求, 其在一些重要点上又能取得较大的导数, 提高学习速度. 文献[14]提出了参数可调的激活函数, 以增加神经网络的非线性映射能力. 文献[15]在 Sigmoid 函数基础上引入了指数和幅度因子, 兼顾了响应的快速性以及平稳性, 具有良好的逼近能力和滤波效果. 上面介绍的激活函数都是十分常见的, 但这些远不够涵盖现存的激活函数.

4 基于 MNIST 数据集的 CNN 实验

卷积神经网络 (Convolutional Neural Network, CNN) 是深度学习算法在图像处理领域的一个应用, 它的特殊性体现在两个方面, 一是神经元间的非全连接性; 二是网络中权值采用共享机制. 本节通过 CNN 实验来验证不同激活函数对深度神经网络的影响. 采用的数据集为 MNIST 数据集, 它是美国中学生手写阿拉伯数值集, 共有 60 000 个 28×28 维 0–9 的手写数字, 其中 50 000 个样本作为训练集, 10 000 个样本作为测试集.

实验在 Ubuntu 系统的 Caffe 框架下进行, 采用 Python 语言进行编程. 实验分析所用的机器配置为: 处理器 CPU 为 Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz 2.59 GHz, 内存 8.00 GB, 操作系统 Ubuntu 14.04, GPU NVIDIA GeForce GTX 950M.

实验采用 caffe 框架中的 Lenet-5 模型. Caffe 中 activation function 的形式直接决定了其训练速度以及随机梯度下降 (SGD) 算法的求解. 针对 Sigmoid、Tanh、ReLU、PReLU、ELU 5 种激活函数, 实验总共设计了 5 次实验, 每次实验采用不同的激活函数. 5 次实验中激活函数是唯一变量, 其余的网络文件以及网络配置文件都相同. 这样设计是为了在同一网络, 同一数据集下来研究不同激活函数的特性以及它们对网络的影响. 每次实验总共对训练集进行了 10 000 次训练, 每 100 次训练输出一次损失函数, 每 500 次使用测试集的数据进行一次测试.

为了观察采用不同激活函数的网络表现出的性能, 对每次实验中的测试精度和损失函数进行统计, 统计对比结果如表 1, 表 2 所示. 之所以要对损失函数进行统计是因为损失函数是衡量数据属于某个类别的概率, 损失函数越小说明网络收敛越快, 而测试精度则是直观反映网络性能的量.

从表 2 看出, 采用 PReLU 激活函数的网络取得了最高的准确率—0.9916, 说明 PReLU 引入了可训练参数α后不仅解决了梯度消失问题, 还大大提高了网络的识别率. 同时, 从表 1 中发现 PReLU 不仅是精度最高的而且是收敛最快更快的, 这是因为 PReLU 的输出更接近 0 均值, SGD 更接近 natural gradient.

表 1 激活函数与迭代次数对应的损失函数值 (Loss)

迭代次数 (次)	1000	2000	3000	4000	5000	6000	7000	8000	9000	10 000
Sigmoid	0.0878851	0.061481	0.0649065	0.044663	0.0406572	0.0379906	0.0386696	0.0369739	0.035584	0.0365851
Tanh	0.0612715	0.0452039	0.0438924	0.0325609	0.0322721	0.0310355	0.0313869	0.0291581	0.0288163	0.0288428
ReLU	0.0584978	0.0414583	0.0353684	0.0307202	0.03102	0.0279192	0.0291126	0.0294737	0.0286014	0.0283891
PReLU	0.0664656	0.046216	0.0477542	0.0314019	0.0302377	0.0276379	0.0290702	0.0285754	0.0282759	0.0270029
ELU	0.0671436	0.045097	0.0402251	0.0331068	0.0349309	0.0298122	0.0337334	0.0336384	0.0319684	0.0318717

表 2 激活函数与迭代次数对应的测试精度 (Accuracy)

迭代次数 (次)	1000	2000	3000	4000	5000	6000	7000	8000	9000	10 000
Sigmoid	0.9733	0.9814	0.9791	0.9866	0.9872	0.9887	0.9886	0.9886	0.9890	0.9887
Tanh	0.98	0.9853	0.9856	0.9895	0.9897	0.99	0.9895	0.9895	0.99	0.9909
ReLU	0.9804	0.9877	0.9887	0.9904	0.9895	0.991	0.9902	0.9902	0.9913	0.9907
PReLU	0.9777	0.9841	0.9831	0.9885	0.989	0.9903	0.9904	0.9904	0.9892	0.9916
ELU	0.9769	0.9846	0.9862	0.9887	0.9888	0.9899	0.989	0.989	0.9893	0.9897

其次表现得较好的是 ReLU 和 Tanh 函数. 在实验中, 两者的网络性能相差并不大. 与传统的 Sigmoid 函数相比, ReLU 能够有效缓解梯度消失问题, 从而直接以监督的方式训练深度神经网络, 无需依赖无监督的逐层预训练. 现在很多网络都采用 ReLU 函数, 不仅因为它的优良性能, 还有一个原因是现在大多数网络都是 fine turning 别人的网络, 而 ReLU 是不引入其他参数的激活函数的最选择.

Tanh 在本文的网络模型中表现的十分良好. 与 Sigmoid 不同的是, Tanh 是 0 均值的, 因此实际应用中, Tanh 会比 Sigmoid 具有更快的收敛速度,实验结果也证明了这一点. 但是 Tanh 并没有解决梯度消失问题, 虽然在本实验中 Tanh 和 Sigmoid 都能很好的收敛, 但是在更复杂的网络中还是要谨慎使用.

从实验结果来看, ELU 融合 Sigmoid 和 ReLU 后确实比 Sigmoid 的性能更好, 但是引起的变化并不大. Sigmoid 函数虽然是最早使用的激活函数, 但是从实验结果来看它的性能却是最差的一个, 不管是从精度上还是收敛速度上都不及后面出现的激活函数.

另外, 对各个激活函数层在 forward 和 backward 过程中消耗的时间进行了统计, 如图 9 所示.

从图 9 可以看出, Sigmoid、Tanh、ELU 函数由于其函数较复杂, 在正向传播过程中耗时较多. PReLU 由于其引入了额外的参数进行训练, 在反向传播过程中有较大的耗时. 虽然 PReLU 在实验所用网络中收敛速度很快, 但当网络越复杂时, 其耗时表现得越明显. ReLU 由于其函数计算简单, 求导容易, 在耗时方面表现出了较好的性能. 再结合上面的分析, ReLU 在训练过程中也能取得较好的准确率, 这就是为什么 ReLU 在激活函数领域始终占据主导地位的原因之一.

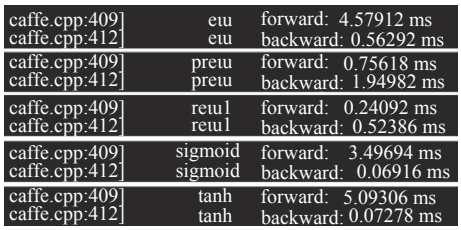

图 9 激活函数 forward/backward 耗时统计图

通过以上实验对比可以看出, 激活函数对一个神经网络的重要程度, 小则影响网络的收敛速度和测试精度, 严重则可能导致梯度消失, 使网络无法训练. 近年, 不仅有创新性的激活函数推出, 许多改进的激活函数也取得了很好的效果. 面对众多的成果, 如何做出恰当的选择目前尚未有统一定论, 仍需依靠实验指导.

5 结语

本文对深度神经网络的重要影响因素激活函数进行了深入研究, 并通过实验分别用 Tank、ReLU、PReLU、ELU 代替原始网络中的 Sigmoid 函数, 取得了更好的结果. 然后通过表格对比方式对采用不同激活函数的网络性能进行了比较. 激活函数的发展对深度神经网络的进一步应用起着至关重要的作用, 其也会是深度神经网络研究中的重点.

参考文献

1 Lecun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015,

521(7553): 436–444.
2 Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proceedings of the 25th International Conference on Neural Information Processing Systems. Lake Tahoe, NV, USA. 2012. 1097–1105.
3 Lecun Y, Boser B, Denker J S, *et al*. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1989, 1(4): 541–551.
4 Chapados N, Bengio Y, Vincent P, *et al*. Estimating car insurance premia: A case study in high-dimensional data inference. Advances in Neural Information Processing Systems. Vancouver, Canada. 2001. 1369–1376.
5 Nair V, Hinton GE. Rectified linear units improve restricted Boltzmann machines. Proceedings of the 27th International Conference on Machine Learning. Haifa, Israel. 2010. 807–814.
6 He KM, Zhang XY, Ren SQ, *et al*. Delving deep into rectifiers: Surpassing human-level performance on imageNet classification. Proceedings of 2015 IEEE International Conference on Computer Vision. Santiago, Chile. 2015. 1026–1034.
7 Xu B, Wang NY, Chen TQ, *et al*. Empirical evaluation of rectified activations in convolutional network. Computer Science, eprint arXiv: 1505.00853, 2015.
8 Goodfellow IJ, Warde-Farley D, Mirza M, *et al*. Maxout networks. ICML 2013. Atlanta, GA, USA. 2013. 1319–1327.
9 Clevert D, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (ELUs). eprint arXiv: 1511.07289, 2015.
10 Ramachandran P, Zoph B, Le QV. Searching for activation functions. ICLR 2018. Vancouver, Canada. 2017.
11 Shang Wl, Sohn K, Almeida D, *et al*. Understanding and improving convolutional neural networks via concatenated rectified linear units. Proceedings of the 33rd International Conference on Machine Learning. New York, NY, USA. 2016. 2217–2225.
12 Li Y, Fan CX, Li Y, *et al*. Improving deep neural network with multiple parametric exponential linear units. Computer Science, eprint arXiv: 1606.00305, 2016.
13 姜天戟, 袁曾任. 新激活函数下前馈型神经网络及其在天气预报中的应用. 信息与控制, 1995, 24(1): 47–51.
14 Carrasco-Robles M, Serrano L. A novel minimum-size activation function and its derivative. IEEE Transactions on Circuits and Systems II: Express Briefs, 2009, 56(4): 280–284.
15 邵星灵, 王宏伦. 基于改进 sigmoid 函数的非线性跟踪微分器. 控制理论与应用, 2014, 31(8): 1116–1122.