基金项目论文

深度神经网络代价函数选择与性能评测研究

赵 宏[1,2*]，郭万鹏[1,2]

（1. 兰州理工大学计算机与通信学院，甘肃 兰州 730050；2. 兰州理工大学信息中心，甘肃 兰州 730050）

摘 要：深度神经网络在诸多领域取得重大突破，已经成为推动人工智能发展的主要力量，但其模型在训练过程中需要使用随机梯度下降算法在大量的训练数据上进行长时间的计算，其中，代价函数和激活函数的选择对模型的训练效率有很大的影响。利用概率论对模型训练中常用的二次代价函数和交叉熵函数进行理论推导，揭示了两者在模型训练过程中对参数寻优的影响，并给出了它们与不同激活函数的组合效果。经过 TensorFlow 平台验证，表明优选的代价函数和激活函数组合可以减少参数寻优的迭代次数从而加快模型的训练过程。

关键词：深度神经网络；交叉熵代价函数；二次代价函数；激活函数；TensorFlow

中图分类号：TP183 **文献标识码：**A **DOI：**10.3969/j.issn.1003-6970.2018.01.004

本文著录格式：赵宏，郭万鹏. 深度神经网络代价函数选择与性能评测研究[J]. 软件，2018，39（01）：14–20

Selection and Evaluation of Cost Function in Deep Neural Network

ZHAO Hong[1,2*], GUO Wan-peng[1,2]

(1. *School of Computer and Communication, Lanzhou University of Technology, Lanzhou,* 730050, *China*;
2. *Information Center, Lanzhou University of Technology, Lanzhou,* 730050, *China*)

【Abstract】: Deep neural network has made the breakthrough in many fields as the main factor to promote the progress of artificial intelligence. However, the training process of its model needs cost long time on the huge training data using the stochastic gradient descent algorithm, in which, the cost and activation function has a profound influence in the training efficiency. The influence of cost and activation function in the training efficiency is disclosed by the derivation using the probability theory, and the combination of different cost and activation function is discussed. The tests in TensorFlow platform show that the optimal combination of cost and activation function can reduce the number of iteration to optimize the parameters in the training process, and the training process will be shortened.

【Key words】: Deep neural network; Cost-entropy function; Quadratic cost function; Activation function; TensorFlow

0 引言

人工神经网络 ANN（Artificial Neural Networks）的研究是人工智能领域的一个重要分支，在对生物神经网络结构及其机制研究的基础上所构建的人工神经网络,使得机器能够直接从数据中提取特征和学习规律，从而使机器具有一定的智能。ANN 的研究最早可以追溯到 1957 年 Frank Rosenblatt 提出的感知机模型，后来出现的反向传播 BP（Back Propagation）算法推动 ANN 进入实用阶段[1]，但由于当时的 ANN 训练数据缺乏、训练方法及技巧不成熟、计算机运算能力不够强大等原因，使得这个时期的 ANN 结构较为简单，可等价为主要用来进行浅层学习的单隐层神经网络。2006 年深度学习领域奠基人 Hinton 教授，根据人脑认知过程的分层特性，提出构建深度神经网络 DNN（Deep Neural Networks），并取得成功[2]。

近年来，深度神经网络在图像识别、语音交互、自然语言处理等领域取得突破，进入实用阶段。与浅层神经网络相比，深度神经网络训练难度更大，其主要原因是梯度下降算法的残差值在网络中逐层传播时变得越来越小，出现梯度消失的问题，使得底层网络由于残差值过小而无法得到有效的训练。从模型训练经验可知，选择合适的代价函数并配合

基金项目：国家自然科学基金(51668043)，赛尔网络下一代互联网技术创新项目(NG1120160311, NG1120160112)

作者简介：赵宏(1971–)，男，甘肃兰州，博士生导师，工学博士，CCF会员，主要研究方向：深度学习、并行与分布式处理；郭万鹏(1992–)，男，甘肃武威，硕士研究生，CCF会员，主要研究方向：深度学习、声纹识别。

相应的激活函数能够明显改善梯度消失的问题，从而加快网络的训练速度。文献[3]对如何提高深度神经网络的学习速率做了大量的研究，并指出交叉熵代价函数能够避免深度神经网络学习速率缓慢的问题，但忽视了激活函数对代价函数性能的影响。文献[4]给出了一种改善深度神经网络训练效果的方法，指出改进激活函数能够提高深度神经网络的训练效果，但结果表明，仅对激活函数的改进，对于提高深度神经网络的学习速率效果并不明显。

利用概率论对模型训练中常用的二次代价函数和交叉熵函数进行理论推导，揭示两者在模型训练过程中对梯度下降算法参数的影响，研究代价函数与激活函数的组合效果，对模型训练过程中的经验进行理论解析并通过实验平台进行验证。

1　基本概念

1.1　独立同分布中心极限定理

定理 1[5]　设随机变量 $X_1,X_2,\cdots,X_n$ 相互独立，服从同一分布，且有 $E(X_i)=\mu,D(X_i)=\sigma^2>0,(i=1,2,\cdots,n)$，其中 $E(X_i)$ 为随机变量 X_i 的数学期望，$D(X_i)$ 为随机变量 X_i 的方差，则随机变量 $X_1,X_2,\cdots,X_n$ 之和 $\sum_{k=1}^{n}X_i$ 的标准化变量 Y_n 如式（1）所示。

$$Y_n=\frac{\sum_{i=1}^{n}X_i-E(\sum_{i=1}^{n}X_i)}{\sqrt{D(\sum_{i=1}^{n}X_i)}}=\frac{\sum_{i=1}^{n}X_i-n\mu}{\sqrt{n}\sigma} \tag{1}$$

其中，对于任意的 x，Y_n 的分布函数 $F_n(x)$ 满足式（2）。

$$\lim_{n\to\infty}F_n(x)=\lim_{n\to\infty}P\left\{\frac{\sum_{i=1}^{n}X_i-n\mu}{\sqrt{n}\sigma}\leqslant x\right\}=\int_{-\infty}^{x}\frac{1}{\sqrt{2\pi}}e^{\frac{t^2}{2}}dt \tag{2}$$

当 n 充分大时，若随机变量 $X_1,X_2,\cdots,X_n$ 服从独立同分布，期望为 μ，方差为 σ^2，则其和 $\sum_{k=1}^{n}X_i$ 近似地服从高斯分布 N，如式（3）所示。

$$\frac{\sum_{i=1}^{n}X_i-n\mu}{\sqrt{n}\sigma}\sim N(0,1) \tag{3}$$

1.2　高斯分布

定义 1[6]　如果随机变量 x 的概率密度函数如式（4）所示，则 x 服从高斯分布 $N(\mu,\sigma^2)$，记为 $x\sim N(\mu,\sigma^2)$，称 x 为高斯随机变量。其中，μ 和 σ 为常数，且 $\sigma>0$。

$$P(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),x\in(-\infty,+\infty) \tag{4}$$

1.3　似然函数

定义 2[7]　样本 $x_1,\cdots,x_n$ 的联合概率函数可看作 θ 的函数，用 $L(\theta;x_1,x_2,\cdots,x_n)$ 表示，称 $L(\theta)$ 为样本的似然函数，简记为 $L(\theta)$，如式（5）所示。

$$\begin{aligned}L(\theta)&=L(\theta;x_1,x_2,\cdots,x_n)=\\&f(x_1;\theta)\cdot f(x_2;\theta)\cdots f(x_n;\theta)=\\&\prod_{i=1}^{n}f(x_i;\theta),(i=1,2,\cdots,n)\end{aligned} \tag{5}$$

2　主要结论

2.1　二次代价函数

二次代价函数[8]的定义如式（6）所示。

$$C(W,b)=\frac{1}{2n}\sum_{x}\|y(x)-\hat{y}(x)\|^2 \tag{6}$$

其中，$C(W,b)$ 表示神经网络的代价函数；n 表示训练样本的大小；W 和 b 分别表示由神经网络权重和偏置构成的矩阵。x 表示由神经网络的输入构成的矩阵；$y(x)$ 表示由神经网络的预期输出构成的矩阵；$\hat{y}(x)$ 表示由神经网络的实际输出构成的矩阵。其中，$\hat{y}(x)$ 的表达式如式（7）所示。

$$\hat{y}(x)=\delta(z)=a \tag{7}$$

$$z=W^Tx+b \tag{8}$$

其中，$\delta(z)$ 表示激活函数。

由式（6）知，训练神经网络的最终目的是获得代价函数 $C(W,b)$ 最小时的权重和偏置，即神经网络的实际输出 $\hat{y}(x)$ 与预期输出 $y(x)$ 的误差最小时的权重和偏置。这是因为假设第 i 个神经元的实际输出为输入特征的线性函数与误差项之和，如式（9）所示。

$$\hat{y}^{(i)}=\theta^Tx^{(i)}+\xi^{(i)} \tag{9}$$

其中，θ 为给定的参数，$\xi^{(i)}$ 为误差项，由定理 1 得

$$\xi^{(i)}\sim N(0,\sigma^2) \tag{10}$$

由式（4）得高斯分布的概率密度函数如式（11）所示。

$$P(\xi^{(i)})=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(\xi^{(i)})^2}{2\sigma^2}\right) \tag{11}$$

在给定参数 θ 后，神经网络的输出 $\hat{y}^{(i)}$ 也服从高斯分布，则 $\hat{y}^{(i)}$ 的概率密度函数如式（12）所示。

$$P(\hat{y}^{(i)} \mid x^{(i)};\theta)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(\hat{y}^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}\right) \quad (12)$$

利用似然函数对参数 θ 进行估计。当 θ 为一定值时，以 θ 为参数的概率 P 取得最大值。由式（5）得

$$L(\theta)=\prod_{i=1}^{n}P(\hat{y}^{(i)} \mid x^{(i)};\theta)= \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(\hat{y}^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}\right) \quad (13)$$

令 $\ell(\theta)=\log L(\theta)$，由式（13）得 $\ell(\theta)$

$$\ell(\theta)=\log\prod_{i=1}^{n}\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(\hat{y}^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}\right)= \sum_{i=1}^{n}\log\left[\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(\hat{y}^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}\right)\right]= \quad (14)$$

$$n\log\frac{1}{\sqrt{2\pi}\sigma}-\sum_{i=1}^{n}\frac{(\hat{y}^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2}$$

令

$$C=\sum_{i=1}^{n}\frac{(\hat{y}^{(i)}-\theta^T x^{(i)})^2}{2\sigma^2} \quad (15)$$

当函数 $\ell(\theta)$ 取得最大值时函数 C 取得最小值。对式（15）进行变换，令

$$y(x)=\theta^T x^{(i)} \quad (16)$$

则由式（15）、式（16）可得：

$$\frac{\partial C}{\partial W}=(a-y(x))\delta'(z)x \quad (17)$$

对式（6）进行链式求导得：

$$\frac{\partial C}{\partial b}=(a-y(x))\delta'(z) \quad (18)$$

其中，假设神经网络的输入 x=1，目标输出 $y(x)$=0，则由式（17）、式（18）知，激活函数的导函数 $\delta'(z)$ 是影响权重和偏置学习速率的重要因素。

引入交叉熵代价函数能够消除 $\delta'(z)$ 对权重和偏置学习速率的影响。

2.2 交叉熵代价函数

假设 $y=y_1,y_2,\cdots y_n$ 是神经元上预期输出值，$a_1^L,a_2^L,\cdots a_j^L$ 是神经元上实际输出值，定义多元神经元上交叉熵代价函数如式（19）所示。

$$C(W_{jk}^L,b_j^L)= -\frac{1}{n}\sum_x\sum_j[y_j\ln a_j^L+(1-y_j)\ln(1-a_j^L)] \quad (19)$$

其中，W_{jk}^L 表示第 L 层上第 j 个神经元与第 k 个神经元之间的权重；b_j^L 表示第 L 层上第 j 个神经元的偏置；x 表示神经元的输入。n 表示训练样本的大小。引入交叉熵代价函数可以避免 $\delta'(z)$ 导致的神经网络学习速率缓慢的问题，这是因为在交叉熵代价函数中消除了参数 $\delta'(z)$ 的影响。由式（17）、式（18）得：

$$\frac{\partial C}{\partial W_{jk}^L}=(a_j^L-y_j)a_k^{L-1}, \quad (20)$$

$$\frac{\partial C}{\partial b_j^L}=(a_j^L-y_j), \quad (21)$$

由式（7）得：

$$\delta'(z)=\delta(z)(1-\delta(z))=a_j^L(1-a_j^L) \quad (22)$$

由式（6）得：

$$\frac{\partial C}{\partial b_j^L}=\frac{\partial C}{\partial a_j^L}\delta'(z) \quad (23)$$

由式（21）、式（22）、式（23）得：

$$\frac{\partial C}{\partial a_j^L}=\frac{a_j^L-y_j}{a_j^L(1-a_j^L)} \quad (24)$$

对式（24）求积分得：

$$\int\frac{a_j^L-y_j}{a_j^L(1-a_j^L)}da_j^L=\int\frac{a_j^L-y_j}{a_j^L}\cdot\frac{1}{1-a_j^L}da_j^L= \int\left(1-\frac{y_j}{a_j^L}\right)\cdot\frac{1}{1-a_j^L}da_j^L= \int\frac{1}{1-a_j^L}da_j^L-y_j\int\left(\frac{1}{a_j^L}+\frac{1}{1-a_j^L}\right)da_j^L= \quad (25)$$

$$(1-y_j)\ln(1-a_j^L)-y_j\ln a_j^L+c_1-y_jc_2$$

令 $c=c_1-y_jc_2$，由式（25）化简得：

$$C=(1-y_j)\ln(1-a_j^L)-y_j\ln a_j^L+c. \quad (26)$$

为了得到多元神经网络上整个交叉熵代价函数，对所有训练样本求平均，得式（19）。同时进一步证明了交叉熵能够解决多元神经元上 $\delta'(z)$ 导致学习速率缓慢的真正原因。式（19）中对参数 W_{jk}^L 进行链式求导得：

$$\frac{\partial C}{\partial W_{jk}^L}=\frac{\partial C}{\partial a_j^L}\cdot\frac{\partial a_j^L}{\partial z_j^L}\cdot\frac{\partial z_j^L}{\partial W_{jk}^L}. \quad (27)$$

由式（19）得：

$$\frac{\partial C}{\partial a_j^L}=-\frac{1}{n}\sum_x\sum_j\left(\frac{y_j}{a_j^L}+\frac{y_{j-1}}{1-a_j^L}\right)= -\frac{1}{n}\sum_x\sum_j\left(\frac{y_j\cdot(1-a_j^L)+a_j^L(y_j-1)}{a_j^L(1-a_j^L)}\right)=$$

$$-\frac{1}{n}\sum_x\sum_j\frac{y_j-a_j^L}{a_j^L(1-a_j^L)} \tag{28}$$

$$\frac{\partial a_j^L}{\partial z_j^L}=\delta'(z_j^L) \tag{29}$$

$$\frac{\partial z_j^L}{\partial W_{jk}^L}=\frac{\partial\left(\sum_k W_{jk}^L a_k^{l-1}+b_j^L\right)}{\partial W_{jk}^L}=a_k^{L-1} \tag{30}$$

由式（28）、式（29）、式（30）得：

$$\frac{\partial C}{\partial W_{jk}^L}=-\frac{1}{n}\sum_x\sum_j\frac{y_j-a_j^L}{a_j^L(1-a_j^L)}\cdot\delta'(z_j^L)a_k^{L-1} \tag{31}$$

由式（22）、式（31）得：

$$\frac{\partial C}{\partial W_{jk}^L}=\frac{1}{n}\sum_x\sum_j(a_j^L-y_j)\cdot a_k^{L-1} \tag{32}$$

式（19）中对参数 b_j^L 进行链式求导得

$$\frac{\partial C}{\partial b_j^L}=\frac{\partial C}{\partial a_j^L}\cdot\frac{\partial a_j^L}{\partial z_j^L}\cdot\frac{\partial z_j^L}{\partial b_j^L} \tag{33}$$

同理可得：

$$\frac{\partial C}{\partial b_j^L}=\frac{1}{n}\sum_x\sum_j(a_j^L-y_j) \tag{34}$$

由式（32）、式（34）知，交叉熵代价函数消除了参数 $\delta'(z)$，解决了多元神经元上 $\delta'(z)$ 导致学习速率缓慢的问题。

通过对二次代价函数和交叉熵代价函数的推导可知，二次代价函数与非线性激活函数结合会导致权重和偏置的学习速率减慢，其真正原因是当 $\delta'(z)$ 的值趋于 0 时，权重和偏置的学习速率受到抑制。引入交叉熵代价函数后，消除了 $\delta'(z)$ 导致学习速率缓慢的问题，从而提高了权重和偏置的学习效率。另外，由式（20）、式（21）知，当神经网络的输入为一定值时，参数的学习速率会随输出误差的变化而变化，当输出误差较大时权重和偏置的学习速率也越快，这也是引入交叉熵代价函数后能够提高神经网络学习速率的重要原因。

由式（17）、式（18）知，不同激活函数导致代价函数学习速率的不同。因此，需要进一步分析代价函数和激活函数组合后的神经网络性能。

2.3　激活函数

神经网络中常用的激活函数分别为 Sigmoid、Tanh、ReLu 和 PReLu。

2.3.1　Sigmoid 激活函数

Sigmoid 函数表达式如式（35）所示，函数值在 0 到 1 之间，如图 1 所示。Sigmoid 激活函数具有双向饱和的特性，当 $x\to\infty$ 时，$\sigma'(x)=0$，具有这种性质的激活函数称为软饱和激活函数[9]，Sigmoid 激活函数的软饱和性是导致神经网络学习速率缓慢的重要原因。

$$\sigma(x)=\frac{1}{1+e^{-x}} \tag{35}$$

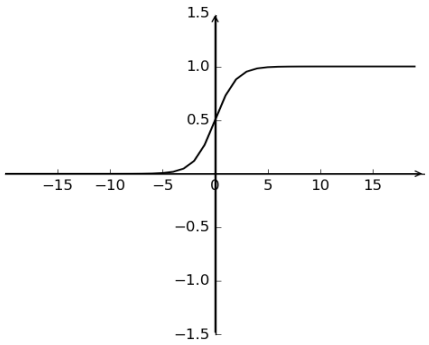

图 1　Sigmoid 函数
Fig.1　Sigmoid function

2.3.2　Tanh 激活函数

Tanh 函数与 Sigmoid 函数具有相似的性质，是典型的 S 型函数，表达式如式（36）所示，函数值在-1 到 1 之间，如图 2 所示。相比 Sigmoid 函数，Tanh 函数延迟了饱和期。对比图 1 和如图 2 可知，Tanh 函数与 Sigmoid 函数一样具有软饱和的特性，同样会陷入局部最优解，使神经网络训练难度加大，但 Tanh 函数性能优于 Sigmoid 函数。

$$\tanh(x)=\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} \tag{36}$$

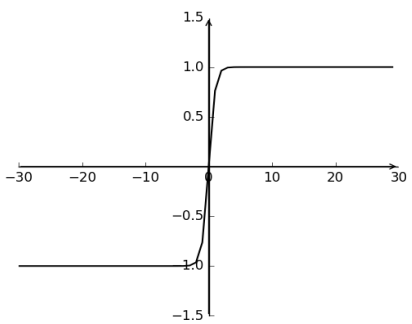

图 2　Tanh 函数
Fig.2　Tanh function

2.3.3　ReLu 激活函数

Relu 函数表达式如式（37）所示，函数值为 0 或 x，如图 3 所示。Relu 激活函数避免了 S 型激活函数饱和区对神经网络收敛速度的限制[10]，但是 Relu 激活函数在训练时非常脆弱，流经 ReLu 神经元的一个大梯度导致权重更新后该神经元接收到的

任何数据点都不会再被激活。

$$y=\begin{cases}0 & x\leqslant 0\\ x & x\geqslant 0\end{cases}. \tag{37}$$

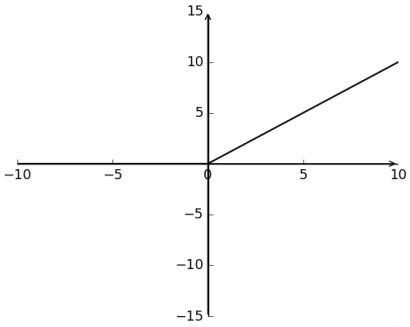

图 3 ReLu 函数
Fig.3 ReLu function

2.3.4 PReLu 激活函数

PReLu（Parametric ReLu）函数表达式如式（38）所示，函数值根据 x 的不同为斜率不同的直线，如图 4 所示。PReLu 激活函数在 ReLu 激活函数的基础上作了进一步改进，当 x 为负值时其导数不再为 0。

$$y=\begin{cases}x_i & x_i>0\\ a_i x_i & x_i\leqslant 0\end{cases}. \tag{38}$$

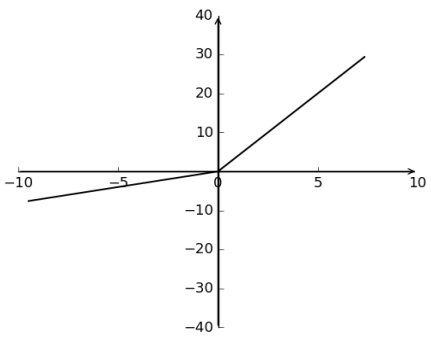

图 4 PReLu 函数
Fig.4 PReLu function

3 实验分析

3.1 CNN 模型原理

卷积神经网络 CNN（Convolutional Neural Network）作为神经网络最常见的形式，使用代价函数和激活函数的组合来提高网络的训练效率和精度。CNN 通过激活函数映射网络的输出，利用代价函数对网络的输出误差进行评价，进而使用反向传播算法和随机梯度下降算法不断提高网络的精度。同时 CNN 的局部感知域、权值共享和池化层等三个主要特征[11]对大型图像的处理具有出色的表现。基于 Hubel 和 Wiesel 早期对猫初级视觉皮层（V1）的研究，CNN 基本结构[12]如图 5 所示，由输入层、卷积层、池化层、全连接层和输出层组成。其中，卷积层和池化层是卷积神经网络特有的属性。卷积层通过卷积操作提取输入的不同特征，池化层对卷积层提取的特征进行池化处理，保留最有用特征，减少神经元数量，避免网络发生过拟合，最后利用 softmax 分类器进行分类，并通过全连接层输出结果。

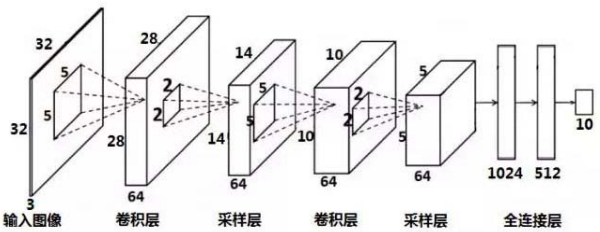

图 5 卷积神经网络的模型图[13]
Fig.5 The model of convolution neural network

3.2 多通道卷积操作

卷积层是 CNN 核心网络层，卷积层基于图像空间的局部相关性，利用卷积提取图像的局部特征，如图 6 所示。

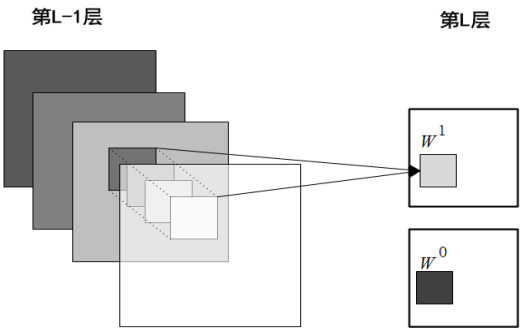

图 6 多通道卷积网络结构[14]
Fig.6 The network structure of multi-channel convolutional

第 L 层特征的计算公式如式（39）所示。

$$z=\sum_j W_{jk}^{L-1}x+b_j^{L-1} \tag{39}$$

其中，z 表示第 L 层特征的输入；W_{jk}^{L-1} 表示第 $L-1$ 层的第 j 个神经元与第 k 个神经元之间的权重；b_j^{L-1} 表示第 L-1层上第 j 个神经元的偏置；x 表示第 $L-1$ 层的输入。

输出图像特征的计算如式（40）所示[14]。

$$L\times H=((L_j-l)/s+1)\times((H_j-h)/s+1) \tag{40}$$

其中，$L\times H$ 表示输出图像尺寸；$L_j\times H_j$ 表示第 j 个神经元上输入图像尺寸；$l\times h$ 表示滤波器尺

寸；s 表示滑动步长。

3.3　池化操作

池化操作是对卷积层提取的特征进行处理，其目的是降低卷积层输出的特征向量，避免网络发生过拟合，同时改善网络训练效果。理论上，可以用所有提取到的特征去训练分类器，但这样做面临计算量的挑战。例如，对于一个 96×96 像素的图像，假设已经学习得到了 400 个定义在 8×8 输入上的特征，每一个特征和图像卷积都会得到 $(96-8+1)\times(96-8+1)=7921$ 维的卷积特征，由于有 400 个特征，所以每个样例都会得到 7921×400=3168400 维的卷积特征向量，训练这样一个超过 3000000 特征输入的分类器容易出现过拟合。根据图像的“静态性”属性，池化操作对卷积层进行如图 7 所示的处理，可以有效防止特征过多造成过拟合的问题，其池化的主要策略有 mean-pooling、max-pooling、stochastic-pooling 等。

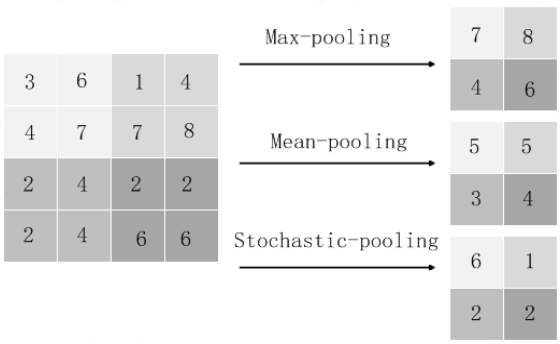

图 7　池化过程
Fig.7　Pooling process

3.4　模型构建

通过对 CNN 模型原理的分析，利用 Google 深度学习平台 TensorFlow 构建了以 CNN 模型为基础的深度神经网络，其中，网络的构建采用了代价函数和激活函数不同组合的方式，并利用 notMNIST 数据作为实验数据集。notMNIST 数据集是一组按字符分类的图像文件夹，有 A、B、C、D、E、F、G、H、I、J 共 10 个文件夹，其可视化后的效果如图 8、图 9 所示，其中，训练数据集中含有部分非字符的噪声图像，如图 9 所示。

3.5　实验结果

表 1、表 2、表 3、表 4 分别为 Sigmoid、Tanh、Relu 和 PReLu 为激活函数时与二次代价函数和交叉熵代价函数的组合效果，表 5 为实验中采用的超参数值。

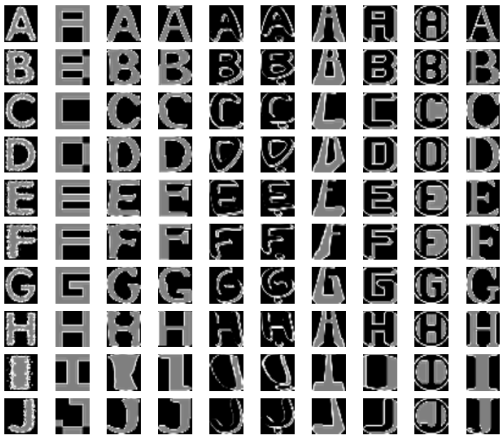

图 8　notMNIST 测试数据集
Fig.8　Test dataset of notMNIST

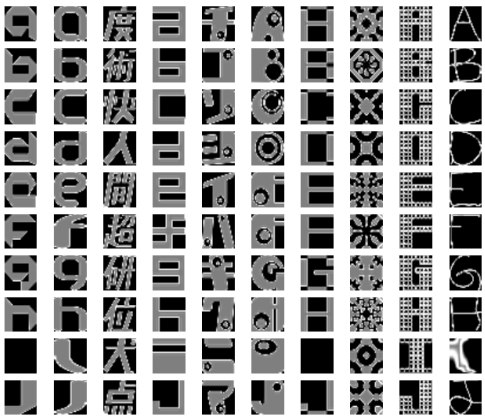

图 9　notMNIST 训练数据集
Fig.9　Training dataset of notMNIST

表 1　Sigmoid 作为激活函数时性能对比
Tab.1　Performance comparison while Sigmoid as the activation function

代价函数	迭代次数	准确率/%
二次代价函数	800	76.406250
交叉熵代价函数	800	87.500000
二次代价函数	1400	78.156250
交叉熵代价函数	1400	85.156250
二次代价函数	2000	77.156250
交叉熵代价函数	2000	89.718750

表 2　Tanh 作为激活函数时性能对比
Tab.2　Performance comparison while Tanh as the activation function

代价函数	迭代次数	准确率/%
二次代价函数	800	90.756350
交叉熵代价函数	800	93.875000
二次代价函数	1400	89.634250
交叉熵代价函数	1400	92.312500
二次代价函数	2000	93.754650
交叉熵代价函数	2000	95.656250

表 3 ReLu 作为激活函数时性能对比
Tab.3 Performance comparison while ReLu as the activation function

代价函数	迭代次数	准确率/%
二次代价函数	800	92.718750
交叉熵代价函数	800	94.531250
二次代价函数	1400	90.718750
交叉熵代价函数	1400	96.093750
二次代价函数	2000	93.812500
交叉熵代价函数	2000	95.033250

表 4 PReLu ReLu 作为激活函数时性能对比
Tab.4 Performance comparison while PReLu as the activation function

代价函数	迭代次数	准确率/%
二次代价函数	800	93.038450
交叉熵代价函数	800	96.563450
二次代价函数	1400	94.324750
交叉熵代价函数	1400	98.972550
二次代价函数	2000	96.241850
交叉熵代价函数	2000	98.532150

表 5 实验中超参数的具体值
Tab.5 The specific value of the over-parameter in the experiment

参数名	值
学习率	0.025
批处理数据	128
权重衰减值	0.0005
L2 规范化因子	0.1
动量	0.9

3.6 实验结果分析

实验探索了不同代价函数和激活函数组合下模型训练的性能，发现代价函数和激活函数的优选会影响模型的训练效率。通过实验结果的对比发现，线性激活函和非线性激活函数对二次代价函数的性能有不同的效果，究其原因，线性激活函数的导函数是一个常数，对二次代价函数的性能并不产生影响，而 Sigmoid 和 Tanh 等非线性函数容易达到饱和导致梯度消失，所以导致学习速率缓慢。实验也表明系统采用 ReLu 函数以及 ReLu 函数的变体函数 PReLu 与代价函数结合使用，会明显改善训练的效果，这是因为 Relu 函数可以通过简单的零阈值矩阵进行激活，并且不受饱和的影响，但是，Relu 函数在训练时非常脆弱，流经 ReLu 神经元的一个大梯度导致权重更新后该神经元接收到的任何数据点都不会再激活。为了解决这个问题，研究者们发现了 ReLu 函数的一些变体函数，如 PReLu，与交叉熵代价函数结合具有良好的训练效果。最后，实验对超参数的设置也进行了探究，并给出了针对该实验具有良好效果的超参数值。

4 结语

本文利用概率论对模型训练中常用的二次代价函数和交叉熵代价函数进行推导，揭示了两者在模型训练过程中对参数寻优的影响，并给出了它们与不同激活函数组合的效果。实验表明，代价函数与激活函数的优选能够减少训练的迭代次数，从而提高深度神经网络训练的效率。并且发现交叉熵代价函数与 PReLU 激活函数结合具有优秀的效果。

参考文献

[1] 庞荣. 深度神经网络算法研究及应用[D]. 西南交通大学, 2016.

[2] Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks[J]. Science, 2006, 313(5786): 504-507.

[3] Michael A. Nielsen. Neural Networks and Deep Learning[M]. Determination Press, 2015.

[4] 叶小舟, 陶飞飞, 戚荣志, 等. 循环神经网络结构中激活函数的改进[J]. 计算机与现代化, 2016(12): 29-33.

[5] 邹广玉. 混合随机变量序列的几乎处处中心极限定理[D]. 吉林大学, 2013.

[6] Zhao L, Mi D, Sun Y. A novel multitarget model of radiation-induced cell killing based on the Gaussian distribution[J]. Journal of Theoretical Biology, 2017, 420: 135-143.

[7] 王晓红, 李宇翔, 余闯等. 基于β似然函数的参数估计方法[J]. 北京航空航天大学学报, 2016, 42(1): 41-46.

[8] 徐先峰, 冯大政. 一种充分利用变量结构的解卷积混合盲源分离新方法[J]. 电子学报, 2009, 37(1): 112-117.

[9] Bao G, Zeng Z. Analysis and design of associative memories based on recurrent neural network with discontinuous activation functions. Neurocomputing[J]. Neurocomputing, 2012, 77(1): 101-107.

[10] 张尧. 激活函数导向的RNN算法优化[D]. 浙江大学, 2017.

[11] 卷积神经网络在图像分类中的应用研究[D]. 电子科技大学, 2015.

[12] 何鹏程. 改进的卷积神经网络模型及其应用研究[D]. 大连理工大学, 2015吴正文.

[13] 李彦冬, 郝宗波, 雷航. 卷积神经网络研究综述[J]. 计算机应用, 2016, 36(9): 2508-2515.

[14] 张重生 深度学习: 原理与应用实践[M]. 电子工业出版社, 2016.