Computer Science and Application 计算机科学与应用, 2018, 8(11), 1773-1781
Published Online November 2018 in Hans. http://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2018.811196

Hans汉斯

Research on Deep Learning Algorithm about Handwriting Based on the Convolutional Neural Network

Liying Ma

Shanghai DianJi University, Shanghai
Email: 15000315026@163.com

Received: Nov. 7th, 2018; accepted: Nov. 19th, 2018; published: Nov. 26th, 2018

Abstract

Deep learning is a collection of algorithms that are used to solve related problems such as image and text. As an important algorithm for deep learning, convolutional neural network is especially good at image processing field. The convolution neural network extracts the various features of the image through the convolution kernel, and its orders of magnitude are greatly reduced by weight sharing and pooling. In this paper, MINST handwritten database is used as training sample to discuss the reverse propagation mechanism of weight value of convolutional neural network, and the implementation method with MATLAB; In order to obtain the optimal correction parameters and learning rate, the problems of gradient disappearance of activation functions tanh and relu were analyzed and optimized, and the improved activation function was trained.

Keywords

Convolutional Neural Network, Deep Learning, Handwriting Recognition

基于卷积神经网络的手写体深度学习算法研究

马利影

上海电机学院，上海
Email: 15000315026@163.com

收稿日期：2018年11月7日；录用日期：2018年11月19日；发布日期：2018年11月26日

摘　要

深度学习是多层神经网络运用各种学习算法解决图像、文本等相关问题的算法合集。卷积神经网络作为

文章引用: 马利影. 基于卷积神经网络的手写体深度学习算法研究[J]. 计算机科学与应用, 2018, 8(11): 1773-1781.
DOI: 10.12677/csa.2018.811196

深度学习的重要算法，尤其擅长图像处理领域。卷积神经网络通过卷积核来提取图像的各种特征，通过权值共享和池化极大降低了网络需要训练的数量级。本文以MINST手写体数据库为训练样本，讨论卷积神经网络的权值反向传播机制和MATLAB的实现方法；对激活函数tanh和relu梯度消失问题进行分析和优化，对改进后的激活函数进行训练，得出最优的修正参数和学习速率。

关键词

卷积神经网络，深度学习，手写体识别，激活函数

1. 引言

卷积神经网络最早是由 Yann Lecun 提出用于手写数字识别，近年来在语音识别，自然语言处理等方面，均取得重大突破[1]。卷积神经网络在传统神经网络的基础上，最重要的改进是局部感知、权值共享，卷积神经网络中的卷积层有若干个特征地图(Feature Map)构成[2]。每个 feature map 由若干神经元组成，代表着提取出的图像特征，这些神经元通过同一个卷积核(权值)去卷积图像而获得，即这些同一个 feature map 共享同一组权值，权值共享机制大大减少了神经网络需要训练的参数量级，例如识别一个 1000 × 1000 的图像，假设隐藏层有 1 M 即 10^6 个神经元，那么传统神经网络采用全连接方式需要训练的参数个数为 $10^6 \times 1000 \times 1000$，即 10^{12} (2 G)个参数，显然这个计算数量级非常庞大。而采用卷积神经网络，假设同样设置隐藏层有 1 M 个神经元，每个神经元采用 10 × 10 大小卷积核去感知图像的固定区域，那么这种方式需要训练的参数为 $10^6 \times 10 \times 10$，即 10^8 (100 M)个参数，卷积神经网络局部连接的算法，一方面降低了需要训练的参数量级，另一方面更符合人类神经对于图像的认知，即先感知局部，再到全局[3]。

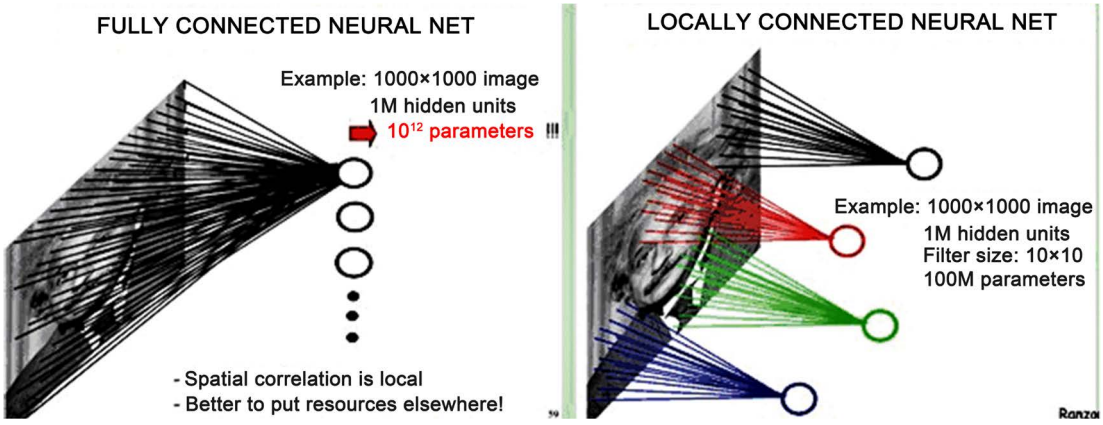

Figure 1. Local perception schematic of convolutional neural network
图 1. 卷积神经网络局部感知示意图

卷积神经网络的第二个重要特性是权值共享[4]，图 1 中采用局部连接仍然需要训练 100 M 个权值参数，如果这 1 M 个神经元共享同一个 10 × 10 的卷积核，那么这 1 M 个神经元仅需要训练 10 × 10 个权值参数，这将极大的优化计算过程，这个 10 × 10 的卷积核相当于一种特征提取器，提取出的 1 M 个神经

元即为一种 feature map，通过不同的卷积核即可提取出图像的不同特征，综合到全局[5]。

由于卷积核比较小，卷积后得到的 feature map 仍然很大，为了降低图像维度，防止过拟合，采用池化即下采样的方法进行处理。池化的方法通常有：最大值下采样(Max-Pooling)与平均值下采样(Mean-Pooling)，例如一个 1000 × 1000 的 feature map 用一个 100 × 100 的采样窗口进行下采样，那么池化后的 feature map 大小为 10 × 10。

例如图 2 所示，卷积层通过不同的卷积核识别出图像的不同特征，经过 6 层的特征提取提取和 3 次池化，最终由全连接层进行分类。

针对传统神经网络的计算复杂、缺少权值共享等问题，本文致力于讨论性能更优的卷积神经网络，在 MATLAB 中实现推导，并优化学习算法。

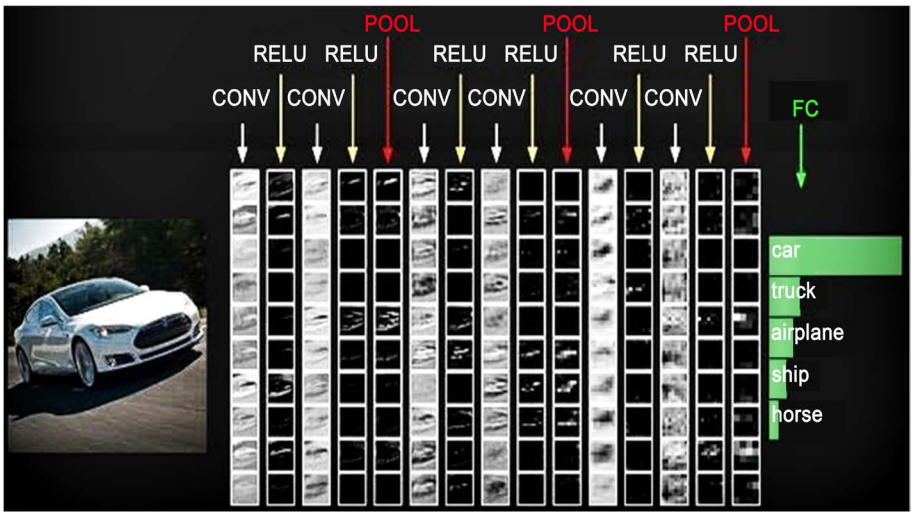

Figure 2. Calculation schematic of convolutional neural network
图 2. 卷积神经网络计算示意图

2. 卷积神经网络的卷积核传递

一个传统神经网络的 BP 过程：先求出各个神经元的灵敏度函数，然后再求得误差函数对权值的导数，这个导数即为权值下降的梯度[6]。

首先，我们需要定义一个优化的目标函数：

$$C=\frac{1}{2}\sum_j\left(y_j-a_j\right)^2 \tag{1}$$

y_j 为输出层*第* j 个神经元的实际值，a_j 为第 j 个神经元的理想值，采用方程(1)的形式定义目标函数，方便后面的求导计算：

$$\frac{\partial C}{\partial a_j}=\left(a_j-y_j\right) \tag{2}$$

首先求输出层神经元的误差：

$$\delta_j^L=\frac{\partial C}{\partial z_j^L}=\frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} \tag{3}$$

z_j^L 代表第 L 层，即输出层第 j 个神经元，a_j^L 为神经元 z_j^L 通过激活函数 $f\left(z\right)$ 后的激活值，那么即可

求出输出层第 j 个神经元的误差为：

$$\delta_j^L = \left(a_j - y_j\right) f'\left(z\right) \tag{4}$$

输出层之前的神经元误差可以由后一层逐层推算，

$$\delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \frac{\partial z_k^{l+1}}{\partial z_j^l} \delta_k^{l+1} \tag{5}$$

其中，z_k^{l+1} 是 $l+1$ 层第 k 个神经元，δ_k^{l+1} 是 $l+1$ 层第 k 个神经元的误差。

由于

$$z_k^{l+1} = \sum_j \omega_{kj}^{l+1} a_j^l + b_k^{l+1} = \sum_j \omega_{kj}^{l+1} f\left(z_j^l\right) + b_k^{l+1} \tag{6}$$

所以

$$\frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} f'\left(z_j^l\right) \tag{7}$$

因此

$$\delta_j^l = \sum_k \omega_{kj}^{l+1} \delta_k^{l+1} f'\left(z_j^l\right) \tag{8}$$

由上面推导过程可以求得每个神经元误差。权值更新需要求代价函数对权值的导数：

$$\frac{\partial C}{\partial \omega^l} = \frac{\partial C}{\partial z^l} \frac{\partial z^l}{\partial \omega^l} = \delta^l \left(a^{l-1}\right)^T \tag{9}$$

代价函数对偏置 b^l 的导数为：

$$\frac{\partial C}{\partial b^l} = \frac{\partial C}{\partial z^l} \frac{\partial z^l}{\partial b^l} = \delta^l \tag{10}$$

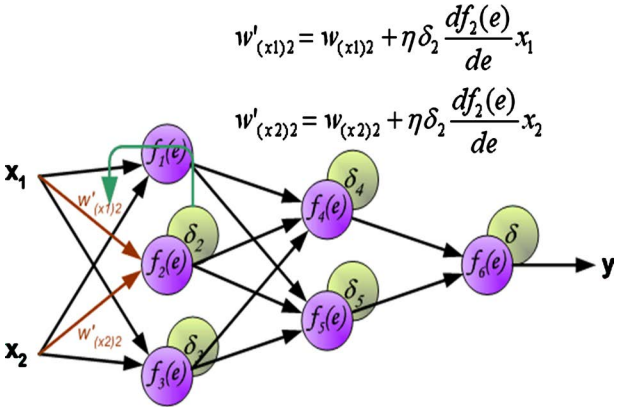

Figure 3. The update process of the neural network weight
图 3. 神经网络的权值更新过程

图 3 即为更新过程，类比于普通神经网络的神经元误差传递，卷积神经网络卷积层神经元误差传递为：

$$\delta_j^l = upsample\left(\delta_j^{l+1}\right) \otimes f'\left(z_j^{l+1}\right) \tag{11}$$

卷积层神经元误差在 MATLAB 中可以通过以下代码实现：

```
for j = 1 : numel(net.layers{l}.a)
net.layers{l}.d{j} = net.layers{l}.a{j} .* (1 - net.layers{l}.a{j}) .* (expand(net.layers{l + 1}.d{j}, [net.layers{l + 1}.scale net.layers{l + 1}.scale 1]) / net.layers{l + 1}.scale ^ 2);
end
```

net.layers{l}.a 为第 1 层神经元，numel(net.layers{l}.a 为卷积层神经元的个数，这里 net.layers{l}.a{j} .* (1 - net.layers{l}.a{j})是对激活函数 f(z)求导。expand(net.layers{l + 1}.d{j}, [net.layers{l + 1}.scale net.layers{l + 1}.scale 1]) / net.layers{l + 1}.scale ^ 2 是上采样操作。

由于池化层神经元没有经过激活函数，神经元误差不需要对激活函数求导，所以池化层神经元误差传递公式为：

$$\delta_j^l = \sum_k \left(\delta_k^{l+1} \otimes rot180\left(\omega_{jk}\right)\right) \tag{12}$$

池化层神经元的误差在 MATLAB 中的实现方式为：

```
for i = 1 : numel(net.layers{l}.a)
z= zeros(size(net.layers{l}.a{1}));
 for j = 1 : numel(net.layers{l + 1}.a)
 z = z + convn(net.layers{l + 1}.d{j}, rot180(net.layers{l + 1}.k{i}{j}), 'full');
 end
 net.layers{l}.d{i} = z;
end
```

numel(net.layers{l}.a)为池化层神经元的个数，numel(net.layers{l + 1}.a)为池化层下一层即卷积层神经元的个数，net.layers{l}.d{i}为池化层第 1 层第 i 个神经元的误差。

在求得池化层核卷积层神经元误差后，下一步需要求得代价函数对网络权值的导数：

由于池化层没有进行权值加成，因此仅对卷积层进行权值梯度计算：

卷积神经网络的权值误差为：

$$\frac{\partial C}{\partial \omega_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial \omega_{jk}^l} = \delta^l * rot\left(a_k^{l-1}\right) \tag{13}$$

$$\frac{\partial C}{\partial b_j^l} = \sum \delta_j^l \Big/ size\left(\delta_j^l\right) \tag{14}$$

卷积核和偏置的误差计算在 MATLAB 的实现方式为：

```
for j = 1 : numel(net.layers{l}.a)
for i = 1 : numel(net.layers{l - 1}.a)
net.layers{l}.dk{i}{j}=convn(flipall(net.layers{l-1}.a{i}),net.layers{l}.d{j},'valid')/size(net.layers{l}.d{j}, 3);
net.layers{l}.db{j} = sum(net.layers{l}.d{j}(:)) / size(net.layers{l}.d{j}, 3);
end
```

numel(net.layers{l}.a 为卷积层 l 神经元的个数，net.layers{l}.dk{i}{j}为第 l 层卷积核 k{i}{j}的误差。net.layers{l}.db{j}为卷积层偏置的误差。

3. 深度学习算法优化

神经网络的非线性建模能力通过神经元的激活函数实现，深度学习使用的广泛的激活函数是 tanh 函

数[7]：

$$f(x)=\frac{2}{1+e^{-2x}}-1 \tag{15}$$

激活函数 tanh 的图像如图 4 所示，可以看出该函数处处可导，但是当输入 $|x|>2$ 时，tanh 导数激活为零，即处于饱和状态，神经元的参数更新的梯度非常小，这就导致神经元的误差传递效率几乎为零。为了避免梯度消失，在 tanh 函数叠加一个系数 kx。

函数 $f(x)=\frac{2}{1+e^{-2x}}-1+kx$，当输入 $|x|>2$ 时，导数不为零，即可增加一个梯度 k。优化后的 tanh 函数如图 5 所示，可以看到梯度明显增加，且不同 k 值会对梯度产生影响。

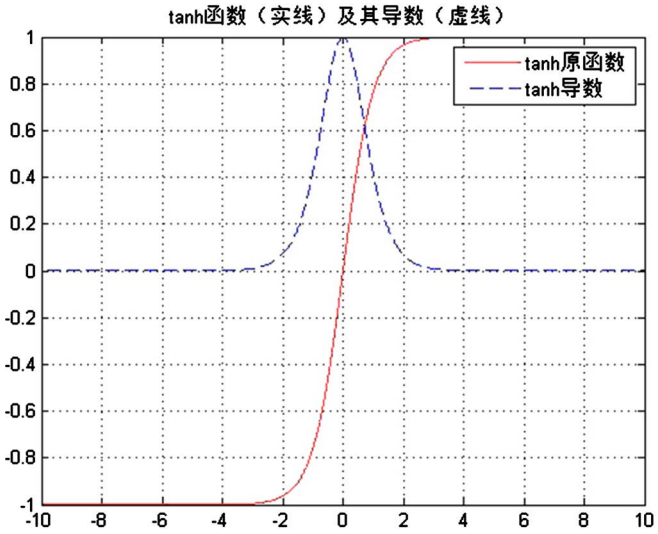

Figure 4. Primitive function and derived function of the tanh
图 4. tanh 的原函数和导数

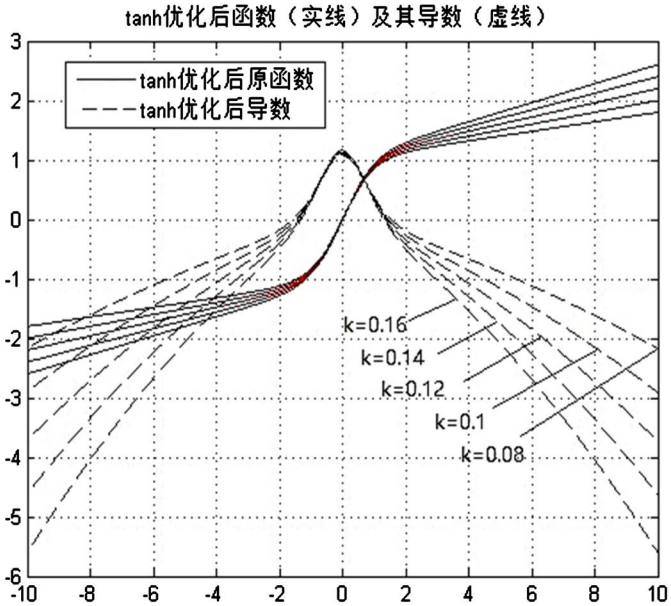

Figure 5. Primitive function and derived function of the optimized tanh
图 5. 优化后 tanh 的原始函数及其导数

图 6 是优化后的激活函数，可以看出当 k = 0.13 时，误差降低速率最快，最终迭代后的误差值为：0.1694，在激活函数优化后，可避免梯度的消失，但是 600 迭代后，残差仍然有 0.1694，需要通过修改学习速率来降低残差。图 7 是学习速率 α 分别取 0.1，1.5，2，3，4，5 时，误差的下降速度。当学习速率取 2 时，误差下降最快，残差值降低至 0.0481。因此，在激活函数参数和学习速率要同时考虑时，才能取到参数的最优值。

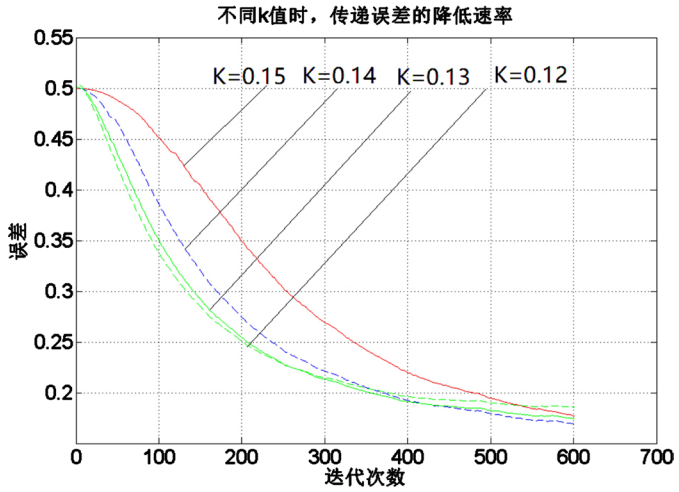

Figure 6. The influence of different k values on the calculation rate of error function
图 6. 不同 k 值对误差函数计算速率的影响

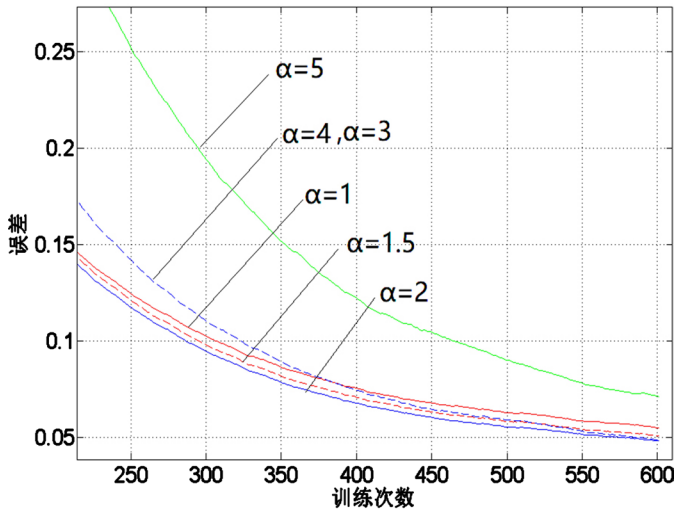

Figure 7. Error reduction in different learning rates
图 7. 不同学习速率的误差降低情况

Tanh 激活函数在求导计算时，涉及指数运算，在处理大规模数据输入时，时间代价较大，为了更加快速地进行误差传递，采用 $f(x)=\max(0,x)$ 即 Relu 函数激活[8] [9]，该激活函数及其导数的图像如图 8 所示。从图中可以看出，当输入大于 0 时，其梯度快速增加，解决梯度消失问题，提供神经网络稀疏表达能力，但是当输入小于 0 时，梯度为 0，在训练过程中权值无法更新，造成部分神经元死亡。为了解决这一问题，需要对 max(0, x)激活函数进行改进。为了避免梯度消失的情况出现，将激活函数小于 0 范围内，增加一个修正系数 α，即：$f(x)=\alpha x, x<0$。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，但是修正系数 α 的选择无法直接确定。为了获得合适的修正系数，对于特定的神经网络，设置同样的修正系数进行训练。图 9 是采用不同的修正系数来训练同一个神经网络的输出结果。$\alpha = 0$ 的情况即为 max(0,*x*)激活函数的训练情况，其误差函数下降速度最缓，部分神经元已经死亡。增加修正系数后，误差函数下降明显提速，说明修正系数有效避免激活函数梯度消失的区间。为了获得更优的修正系数，继续增加修正系数的值，进行训练，从训练结果来看，当系数增加到 0.06，误差函数的下降梯度变缓，因此，通过 6 组实验数据的训练曲线来看，修正系数 α 取 0.05 时，训练效果最佳。

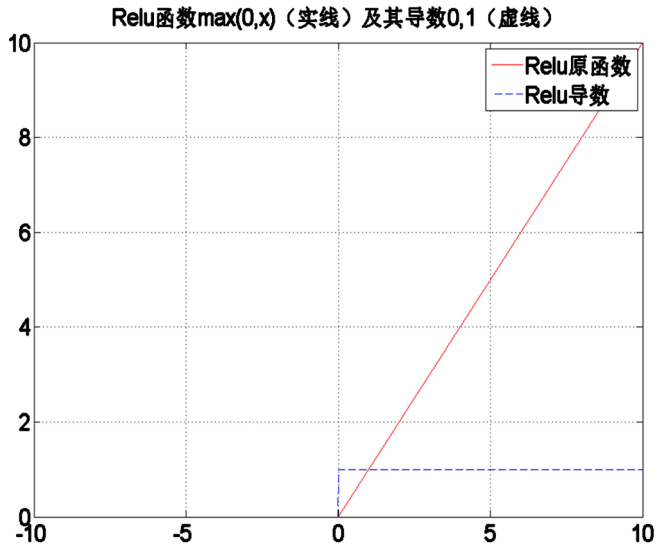

Figure 8. Primitive function and derived function of the Relu
图 8. Relu 的原函数和导数

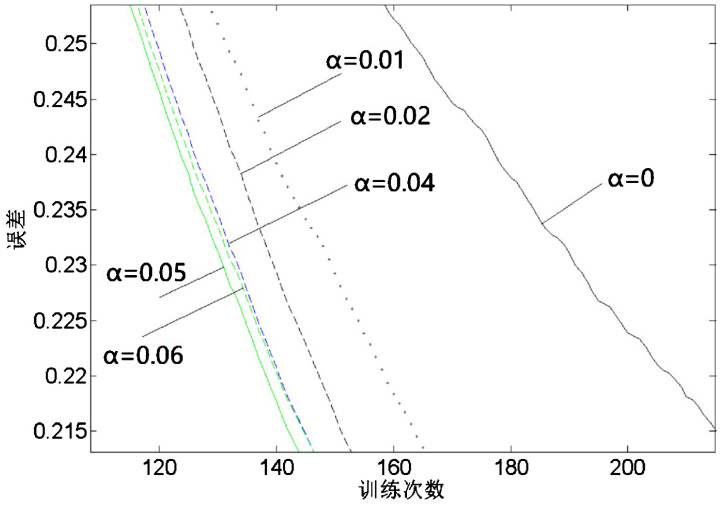

Figure 9. Selection of correction coefficient
图 9. 修正系数选择

4. 总结

讨论深度学习网络的误差传递计算过程，应用 matlab 建立深度学习网络对手写体进行识别，分析激活函数梯度饱和问题，优化激活函数，解决误差传递过程中梯度消失问题，通过计算得出最优的激活函

数。通过对不同学习速率系数的计算，得出最优学习速率系数，在 600 次迭代中，使得神经网络的残差从 0.169 降低到 0.0481。

本文讨论卷积神经网络算法的参数优化和算法的 MATLAB 实现，对以后深度学习尤其是在图像识别方面的应用具有一定的价值。

参考文献

[1] 王心宇, 马良, 蔡瑞. 基于卷积神经网络的图像识别[J]. 工程技术研究, 2018(4): 101-102.
[2] 朱瑞. 基于卷积神经网络的图像识别算法的研究[D]: [硕士学位论文]. 北京: 北京邮电大学, 2018.
[3] 段金宝. 基于深度神经网络的证件图像文本识别方法[D]: [硕士学位论文]. 北京: 北京邮电大学, 2018.
[4] 尚泽元. 基于深度区域卷积神经网络图像识别的研究[J]. 中国战略新兴产业, 2017(44): 151-153.
[5] 文馗. 基于深度学习的图像识别方法研究与应用[D]: [硕士学位论文]. 武汉: 华中师范大学, 2017.
[6] 焦李成, 杨淑媛, 刘芳, 王士刚, 冯志玺. 神经网络七十年: 回顾与展望[J]. 计算机学报, 2016, 39(8): 1697-1716.
[7] 陈雪. 神经网络的图像识别技术及方法分析[J]. 通讯世界, 2016(1): 39-40.
[8] 战国科. 基于人工神经网络的图像识别方法研究[D]: [硕士学位论文]. 北京: 中国计量科学研究院, 2007.
[9] 彭淑敏. 神经网络图像识别技术研究与实现[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2005.