信陽師範学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

第14章 神经网络与深度学习

讲授人：牛言涛

日期：2020年5月19日

目录

CONTENTS

1. 深度学习概述

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

- 深度学习是机器学习的一种，而机器学习是实现人工智能的必经路径。深度学习的概念源于人工神经网络的研究，含多个隐藏层的多层感知器就是一种深度学习结构。
- 深度学习通过组合低层特征形成更加抽象的高层表示属性类别或特征，以发现数据的分布式特征表示。
- 研究深度学习的动机在于建立模拟人脑进行分析学习的神经网络，它模仿人脑的机制来解释数据，例如图像，声音和文本等。
- 区别于传统的浅层学习，深度学习的不同在于：

  (1)强调了模型结构的深度，通常有5层、6层，甚至10多层的隐层节点；

  (2)明确了特征学习的重要性。也就是说，通过逐层特征变换，将样本在原空间的特征表示变换到一个新特征空间，从而使分类或预测更容易。

1. 深度学习概述

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

- 深度学习是一种利用深度神经网络框架的机器学习技术，是一种包含两层以上隐藏层的多层神经网络。
- 深度学习的核心是特征学习，旨在通过分层网络获取分层次的特征信息，从而解决以往需要人工设计特征的重要难题。深度学习是一个框架，包含多个重要算法：
  - Convolutional Neural Networks(CNN)卷积神经网络
  - AutoEncoder自动编码器
  - Sparse Coding稀疏编码
  - Restricted Boltzmann Machine(RBM)限制波尔兹曼机
  - Deep Belief Networks(DBN)深信度网络
  - Recurrent neural Network(RNN)多层反馈循环神经网络

2. 深度神经网络的进化

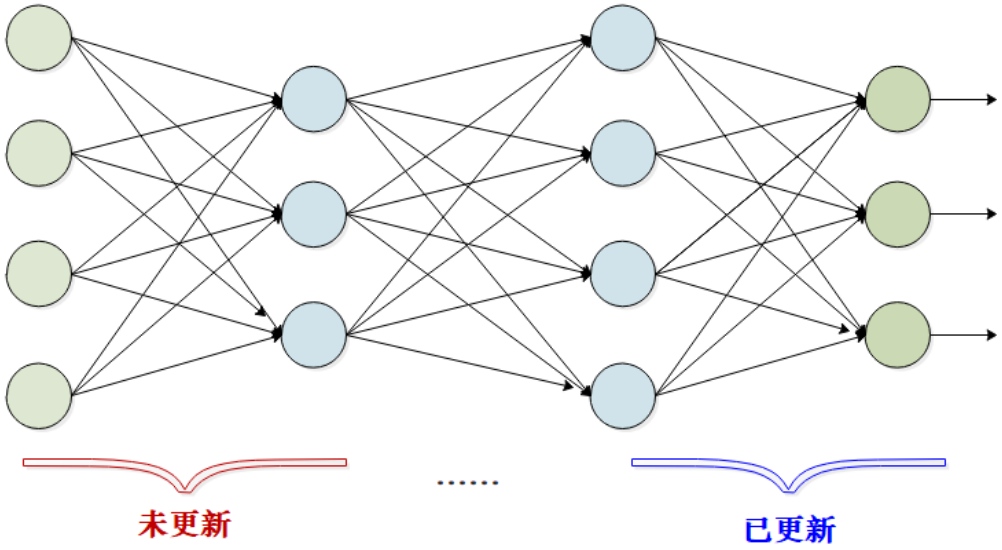
信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

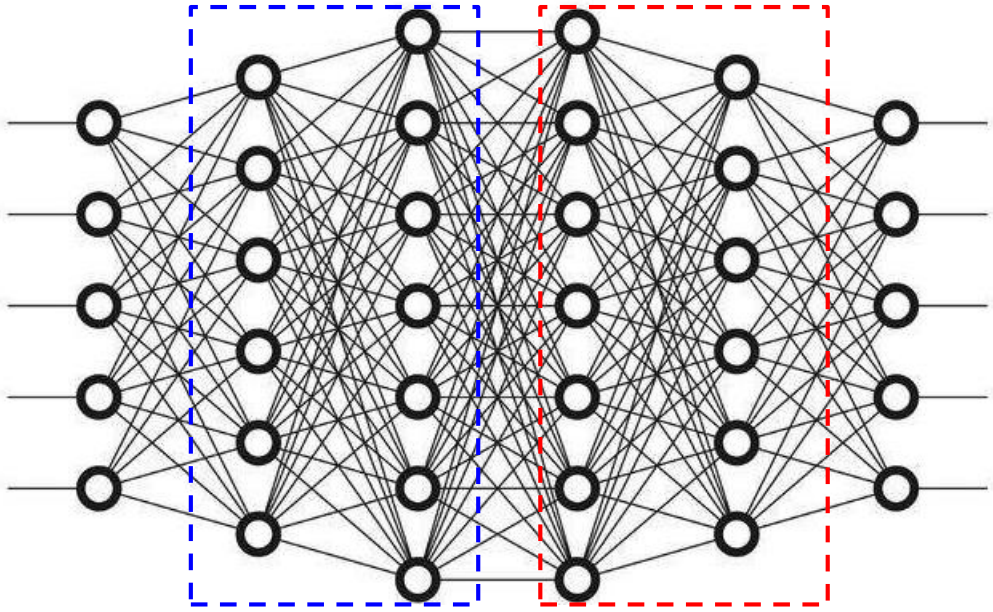
在深度神经网络训练算法中，反向传播算法面临着如下难题：

- 梯度消失：使用ReLU作为激活函数；
- 过拟合：解决方法是Dropout随机丢弃节点；
- 计算量的增加
  - ✓ 权重的数量随着隐藏层数量的增加而呈几何式增长，这会需要更多的训练数据，最终导致需要大量的计算。
  - ✓ 神经网络执行的计算越多，训练所需要的时间就越长，这是神经网络模型在实际开发中需要特别重视的问题。

(1) 梯度消失

- 反向传播算法是将输出误差反向传播至隐藏层来训练神经网络的方法，然而当误差几乎无法到达第一个隐藏层时（梯度消失现象），权重就得不到调整，即输入层附近的隐藏层得不到恰当的训练，增加的隐藏层也就变得毫无意义。
- 换句话说，当梯度消失发生时，接近于输出层的隐藏层由于其梯度相对正常，所以权值更新时也就相对正常，但是当越靠近输入层时，由于梯度消失现象，会导致靠近输入层的隐藏层权值更新缓慢或者更新停滞。这就导致在训练时，只等价于后面几层的浅层网络的学习。





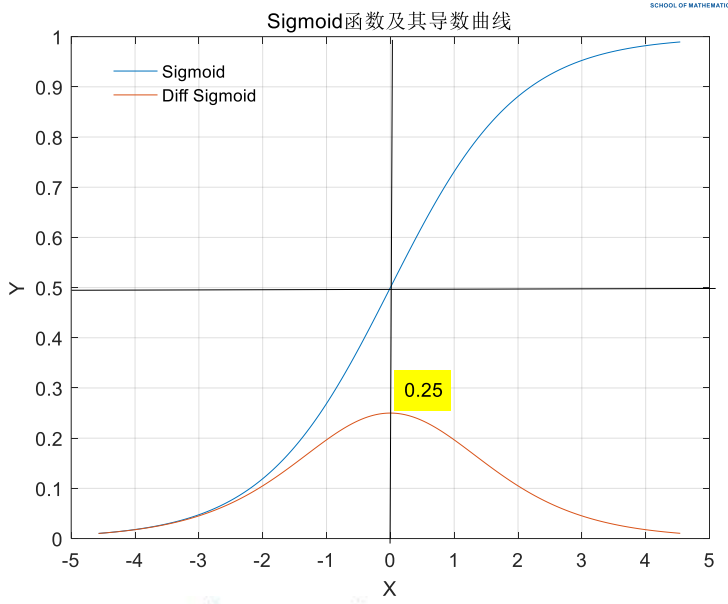
(1) 梯度消失

- 产生消失的梯度问题的原因

- 代价函数C对偏置b1的偏导数的结果计算如下：

$$\frac{\partial C}{\partial b_1}=\sigma'(z_1)\times w_2\times\sigma'(z_2)\times w_3\times\sigma'(z_3)\times w_4\times\sigma'(z_4)\times\frac{\partial C}{\partial a_4}$$

- sigmoid 函数导数的图像，该导数在$\sigma'(0)=1/4$时达到最高



BP算法的基础是导数的链式法则。而sigmoid的导数最大为0.25，且大部分数值都被推向两侧饱和区域，这就导致大部分数值经过sigmoid激活函数之后，其导数都非常小（权重初始通常小于1），多个小于等于0.25的数值相乘，其运算结果很小。随着神经网络层数的加深，梯度后向传播到浅层网络时，基本无法引起参数的扰动，即没有将Loss的信息传递到浅层网络，这就是梯度消失。（梯度爆炸，权重与激活函数的成绩大于1）。

$$\frac{\partial C}{\partial b_1}=\sigma'(z_1)\overbrace{w_2\sigma'(z_2)}^{<\frac{1}{4}}\overbrace{w_3\sigma'(z_3)}^{<\frac{1}{4}}\underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

common terms

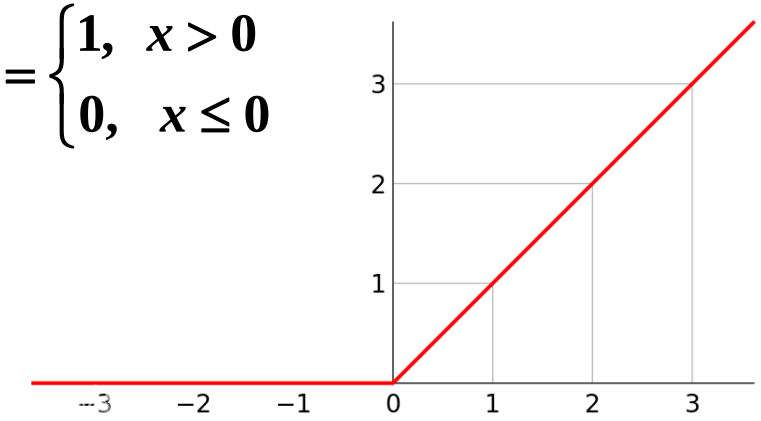
$$\frac{\partial C}{\partial b_3}=\sigma'(z_3)\underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

(1) 梯度消失

- 如何确定是否出现梯度爆炸？如：模型无法从训练数据中获得更新（如低损失）；模型不稳定，导致更新过程中的损失出现显著变化；训练过程中，模型损失变成NaN。
- 解决梯度消失问题的典型方法是将修正线性单元(Rectified Linear Unit, ReLU)函数作为激活函数，它比Sigmoid函数能更好地传递误差。采用 ReLU 激活函数是最适合隐藏层的，是目前使用最多的激活函数。定义：

$$\varphi(x)=\begin{cases}x, & x>0\\ 0, & x\le 0\end{cases}=\max(0,x)$$

$$\varphi'(x)=\begin{cases}1, & x>0\\ 0, & x\le 0\end{cases}$$



relu的主要贡献在于：

1. -- 解决了梯度消失、爆炸的问题

2. -- 计算方便，计算速度快， 加速了网络的训练

同时也存在一些缺点：

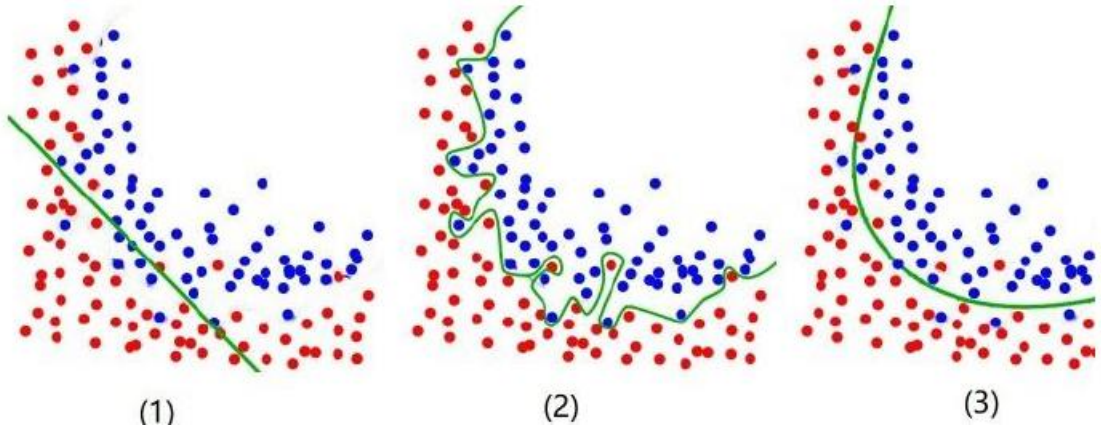
1. 由于负数部分恒为0，会导致一些神经元无法激活（可通过设置小学习率部分解决）

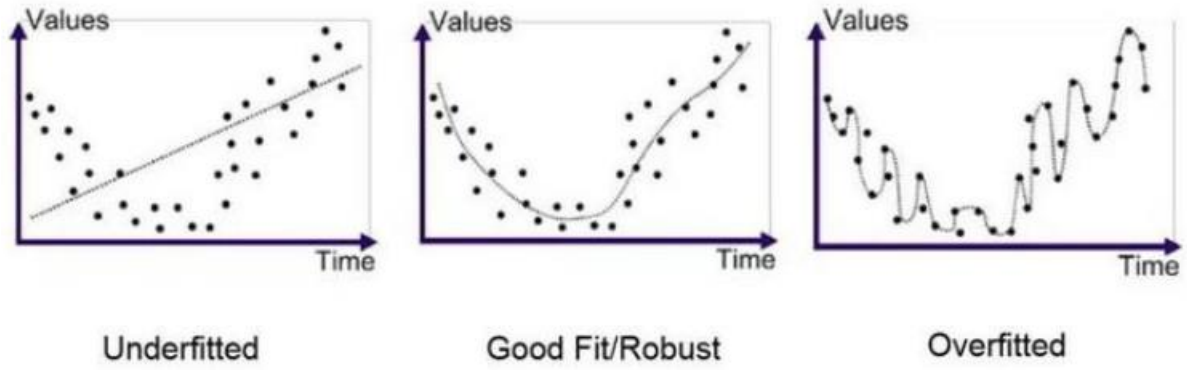
2. 输出不是以0为中心的。

(2) 过拟合

损失函数是用于优化训练数据。然而真实的应用中想要的并不是让模型尽量模拟训练数据的行为，而是希望通过训练出来的模型对未知的数据给予判断。模型在训练数据上的表现并不一定代表了它在未知数据上的表现。

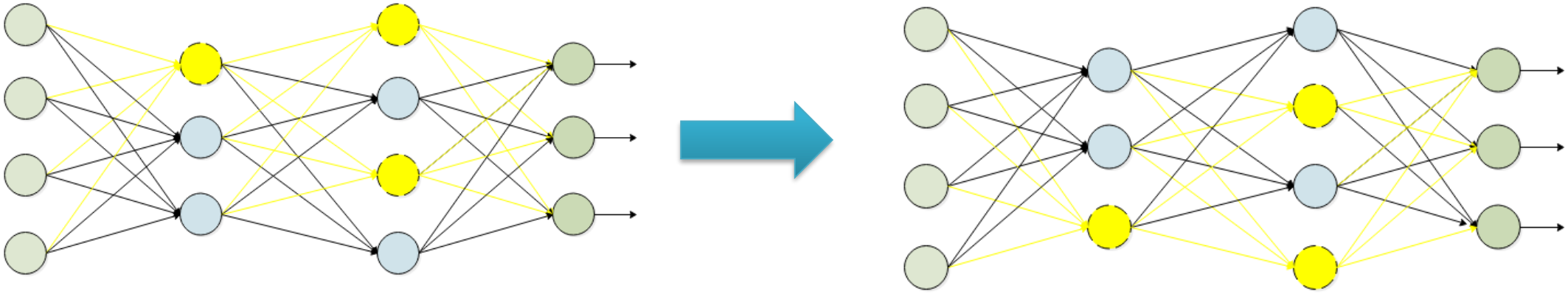
过拟合就是导致这个差距的重要因素，所谓过拟合就是当一个模型过于复杂之后，它可以很好的“记忆”每一个训练数据中随机噪音的部分而忘记了要去“学习”训练数据中的通用趋势。





(2) 过拟合

- 深度神经网络导致过拟合的原因是它含有更多的隐藏层和更多的权重，而从使问题变得更复杂，即复杂模型更容易导致过拟合。
- 最具代表性的解决方案是节点丢弃(dropout)，神经网络只训练那些随机挑选的节点，而不是全部节点。



黄色节点表示随机丢弃的节点

下一次训练

(2) 过拟合

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

- 在训练过程中，节点丢弃算法在持续地转换节点和权重，能有效地防止过拟合。对于隐藏层和输入层节点来说，较为合适的节点丢弃百分比分别约为50%和25%。
- 另一个防止过拟合的方法是，给代价函数增加能够提供权重大小的正则项，在训练的时候限制权值变大，它会尽可能地简化神经网络结构，进而降低过拟合的可能性。

$$Loss = \left(Y - W^T X\right)^2 + \lambda \|W\|^2$$

正则化是通过对网络权重做正则限制过拟合，其中，λ是指正则项系数，如果发生梯度爆炸，权值的范数就会变的非常大，通过正则化项，可以部分限制梯度爆炸的发生。注：事实上，在深度神经网络中，往往是梯度消失出现的更多一些。

- 使用大量训练数据也非常有帮助，能降低特定数据的潜在偏差。这是解决过拟合最有效的方法，只要给足够多的数据，让模型“看见”尽可能多的“例外情况”，它就会不断修正自己，从而得到更好的结果。

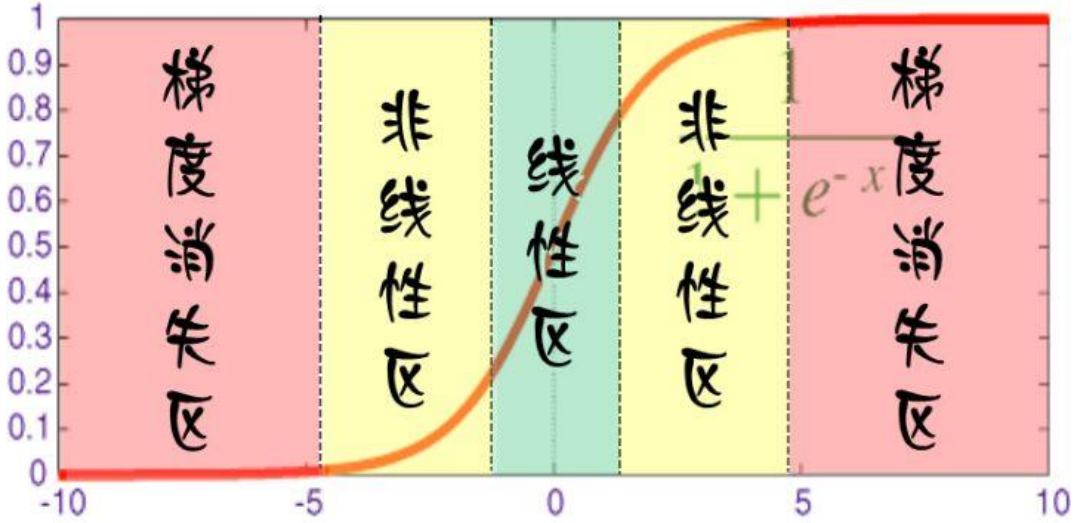
(2) 过拟合

- 如何获取更多数据，可以有以下几个方法：1）从数据源头获取更多数据。2）数据增强（Data Augmentation）：通过一定规则扩充数据。如在物体分类问题里，物体在图像中的位置、姿态、尺度，整体图片明暗度等都不会影响分类结果，就可以通过图像平移、翻转、缩放、切割等手段将数据库成倍扩充。
- 使用合适的模型。过拟合主要是有两个原因造成的：数据太少+模型太复杂。1）减少网络的层数、神经元个数等均可以限制网络的拟合能力；2）训练时间越长，部分网络权值可能越大。在合适时间停止训练，就可以将网络的能力限制在一定范围内。
- 增加噪声 Noise：在输入中加噪声，在权值上加噪声，对网络的响应加噪声
- 防止overfitting三个角度：

  1）数据：augmentation；

  2）网络：简单化网络；

  3）plus：正则，dropout等网络模型的变种。



深度学习算法MATLAB实现—ReLU函数

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

DeepLearning_ReLU.m

```
function [Wh, Wo, Y] = DeepLearning_ReLU(X, labels, nh, alpha, epochs)
% 该函数实现深度学习多分类自适应BP算法，使用ReLU激活函数，针对图片分类
% 输入X是一个三维矩阵，其中第三维表示样本数，labels是正确类别标签向量
% 输入nh是隐藏层神经元数向量，根据向量的长度设置隐藏层数
% 如nh = [100 80 20 10]则包含四层隐藏层的神经网络

    fn = length(unique(labels));  %类别数
    lenh = length(nh);
    %% one-hot编码
    [m, n, num] = size(X);
    D = zeros(num, fn);
    for i = 1:num
        D(i, labels(i)) = 1;
    end

    %% 权重的初始化
    Wh = cell(lenh, 1);
    Wh{1} = 2*rand(nh(1), m*n) - 1; %输入层到隐藏层1
    for i = 2:lenh
        Wh{i} = 2*rand(nh(i), nh(i-1)) - 1; %输入层到隐藏层1
    end
    Wo = 2*rand(fn, nh(end)) - 1; %隐藏层3到输出层

    %% 网络训练
    Error = zeros(1, epochs);
    for i = 1:epochs
        out = zeros(fn, num);
        for k = 1:num
            %% 信号向前传播
            x = reshape(X(:, :, k), m*n, 1);  %每循环一次，读取一个数据
            [v, y] = forwardPropagation(x, Wh, Wo);
            out(:, k) = y{end};
            %% 误差向后传播
            e = D(k, :)' - y{end};  %误差
            [Wh, Wo] = backPropagation(v, y, x, Wh, Wo, e);
        end
        Error(i) = mean(mean((D' - out).^2));
        if mod(i, 10) == 0
            fprintf('第【%d】次训练.....\n', i);
        end
    end

    %% 模型最终输出
    Y = zeros(fn, num);
    for k = 1:num
        x  = reshape(X(:, :, k), m*n, 1);
        [~, y] = forwardPropagation(x, Wh, Wo);
        Y(:, k) = y{end};
    end

    %% 统计识别正确率
    hitNum = 0;
    for i = 1:num
        [~, Index] = max(Y(:, i));
        if Index  == labels(i)
            hitNum = hitNum + 1 ;
        end
    end
```

深度学习算法MATLAB实现—ReLU函数

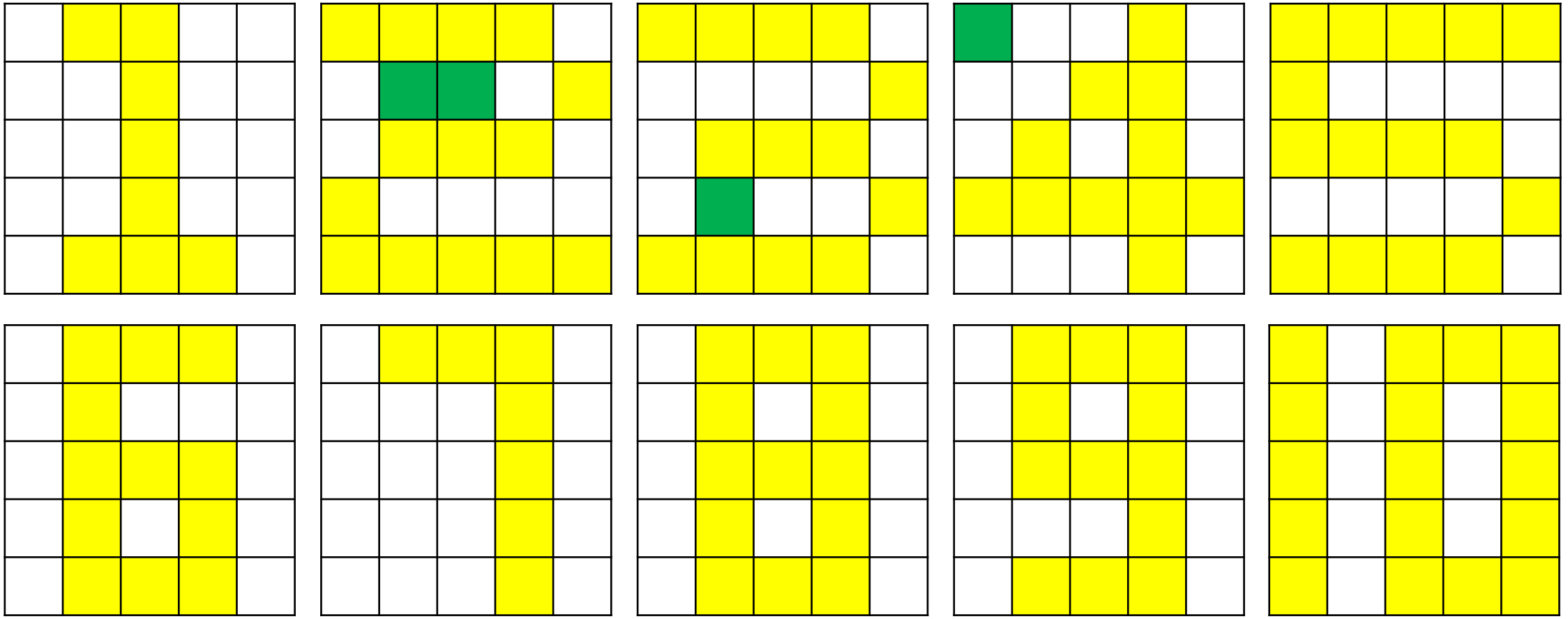
信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

```
59 —      fprintf('识别正确率为：%3.3f。\n',100 * hitNum / num )
60
61 —      %% 误差损失曲线
62 —      plot(Error(1:5:end),'r.-','LineWidth',1)
63 —      grid on
64 —      xlabel('epochs/5'); ylabel('Loss')
65 —      title(strcat('深度学习BP算法（ReLU函数）误差损失曲线 Loss = ',...
66            num2str(Error(end))))
67
68        %% 信号前向传播
69        function [v,y] = forwardPropagation(x,Wh,Wo)
70 —          v = cell(lenh,1);
71 —          y = cell(lenh+1,1);
72 —          v{1} = Wh{1}*x;  %输入层到隐藏层权重和
73 —          y{1} = ReLU(v{1}); %修正线性单元激活函数，隐藏层1输出
74 —          for ifp = 2:lenh
75 —              v{ifp} = Wh{ifp}*y{ifp-1};
76 —              y{ifp} = ReLU(v{ifp});
77 —          end
78            %从隐藏层到输出层,softmax激活函数
79 —          y{lenh+1} = Softmax(Wo*y{lenh});
80 —      end
81
82        %% 误差反向传播
83        function [Wh,Wo] = backPropagation(v,y,x,Wh,Wo,e)
84 —          delta = cell(1,lenh+1);
85 —          ebp = cell(1,lenh+1);
86 —          delta{end} = e;
87 —          ebp{end} = Wo'*delta{end};
88 —          dWo = alpha*delta{end}*y{end-1}'; %delta rule
89 —          Wo  = Wo + dWo; %更新权重，从输入层到隐藏层end-1
90 —          for ibp = lenh:-1:2
91 —              delta{ibp} = (v{ibp} > 0).*ebp{ibp+1};
92 —              ebp{ibp} = Wh{ibp}'*delta{ibp};
93 —              dWh = alpha*delta{ibp}*y{ibp-1}';
94                %更新权重，从隐藏层lenh到隐藏层lenh-1
95 —              Wh{ibp}  = Wh{ibp} + dWh;
96 —          end
97 —          delta{1} = (v{1} > 0).*ebp{2};
98 —          dWh1 = alpha*delta{1}*x';
99 —          Wh{1}  = Wh{1} + dWh1; %更新权重，从隐藏层1到输入层
100 —     end
101
102       %% 激活函数
103       function yr = ReLU(x)
104 —          yr = max(0, x);
105 —      end
106       %% 激活函数
107       function ys = Softmax(x)
108 —          ex = exp(x);
109 —          ys = ex./sum(ex);
110 —      end
111 —  end
```

3. 多分类神经网络示例

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

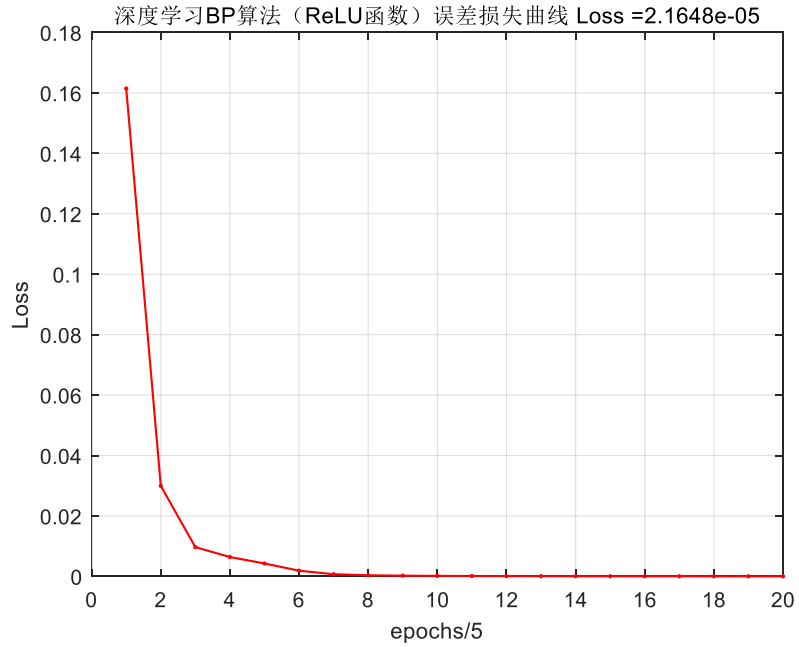
数字识别，如下为1~10十个数字，未加噪声。这里生成15个数字，其中1~5个数字加入了随机噪声。共有类别向量为[1 2 3 4 5 1 2 3 4 5 6 7 8 9 10]。



3. 多分类神经网络示例

```
X(:,:,1) = [0 1 1 0 0;0 0 1 0 0;0 0 1 0 0;0 0 1 0 0;0 1 1 1 0];
X(:,:,2) = [1 1 1 1 0;0 0 0 0 1;0 1 1 1 0;1 0 0 0 0;1 1 1 1 1];
X(:,:,3) = [1 1 1 1 0;0 0 0 0 1;0 1 1 1 0;0 0 0 0 1;1 1 1 1 0];
X(:,:,4) = [0 0 0 1 0;0 0 1 1 0;0 1 0 1 0;1 1 1 1 1;0 0 0 1 0];
X(:,:,5) = [1 1 1 1 1;1 0 0 0 0;1 1 1 1 0;0 0 0 0 1;1 1 1 1 0];
X(:,:,6) = [0 0 1 1 0;0 0 1 1 0;0 1 0 1 0;0 0 0 1 0;0 1 1 1 0];
X(:,:,7) = [1 1 1 1 0;0 0 0 0 1;0 1 1 1 0;1 0 0 0 1;1 1 1 1 1];
X(:,:,8) = [1 1 1 1 0;0 0 0 0 1;0 1 1 1 0;1 0 0 0 1;1 1 1 1 0];
X(:,:,9) = [0 1 1 1 0;0 1 0 0 0;0 1 1 1 0;0 0 0 1 0;0 1 1 1 0];
X(:,:,10) = [0 1 1 1 1;0 1 0 0 0;0 1 1 1 0;0 0 0 1 0;1 1 1 1 0];
X(:,:,11) = [0 1 1 1 0;0 1 0 0 0;0 1 1 1 0;0 1 0 1 0;0 1 1 1 0];
X(:,:,12) = [0 1 1 1 0;0 0 0 1 0;0 0 0 1 0;0 0 0 1 0;0 0 0 1 0];
X(:,:,13) = [0 1 1 1 0;0 1 0 1 0;0 1 1 1 0;0 1 0 1 0;0 1 1 1 0];
X(:,:,14) = [0 1 1 1 0;0 1 0 1 0;0 1 1 1 0;0 0 0 1 0;0 1 1 1 0];
X(:,:,15) = [1 0 1 1 1;1 0 1 0 1;1 0 1 0 1;1 0 1 0 1;1 0 1 1 1];
labels = [1,2,3,4,5,1,2,3,4,5,6 7 8 9 10]';
[Wh,Wo,Y] = DeepLearning_ReLU(X,labels,[50,25],0.01,100);
```

```
>> [Wh, Wo, Y] = DeepLearning_ReLU(X, labels, [50, 25], 0.01, 100);
第【10】次训练.....
第【20】次训练.....
第【30】次训练.....
第【40】次训练.....
第【50】次训练.....
第【60】次训练.....
第【70】次训练.....
第【80】次训练.....
第【90】次训练.....
第【100】次训练.....
识别正确率为：100.000。
```



3. 多分类神经网络示例

```
>> [Wh, Wo, Y] = DeepLearning_ReLU(X, labels, [50, 25], 0.01, 10);
第【10】次训练.....
识别正确率为：100.000。
>> C = Y(:, 6:10)
C =
    0.8753    0.0040    0.0001    0.0141    0.0008
    0.0684    0.7351    0.0082    0.0010    0.0005
    0.0000    0.1290    0.9333    0.0012    0.0016
    0.0001    0.0103    0.0228    0.6886    0.0002
    0.0010    0.0011    0.0020    0.0304    0.9323
    0.0045    0.0000    0.0000    0.0284    0.0018
    0.0004    0.0221    0.0240    0.0388    0.0454
    0.0234    0.0000    0.0003    0.1393    0.0168
    0.0268    0.0002    0.0004    0.0525    0.0005
    0.0000    0.0981    0.0090    0.0058    0.0001
```

从添加噪声的1~5数字训练结果看，两个隐藏层训练10次精度较高，学习速度更快。

```
>> [Wh, Wo, Y] = DeepLearning_ReLU(X, labels, [50, 25, 10, 10], 0.01, 100);
第【10】次训练.....
第【20】次训练.....
第【30】次训练.....
第【40】次训练.....
第【50】次训练.....
第【60】次训练.....
第【70】次训练.....
第【80】次训练.....
第【90】次训练.....
第【100】次训练.....
识别正确率为：60.000。
```

四个隐藏层平均正确率很难超过50%，模型过于复杂。

```
>> [Wh, Wo, Y] = DeepLearning_ReLU(X, labels, [50], 0.01, 100);
第【10】次训练.....
第【20】次训练.....
第【30】次训练.....
第【40】次训练.....
第【50】次训练.....
第【60】次训练.....
第【70】次训练.....
第【80】次训练.....
第【90】次训练.....
第【100】次训练.....
识别正确率为：100.000。
```

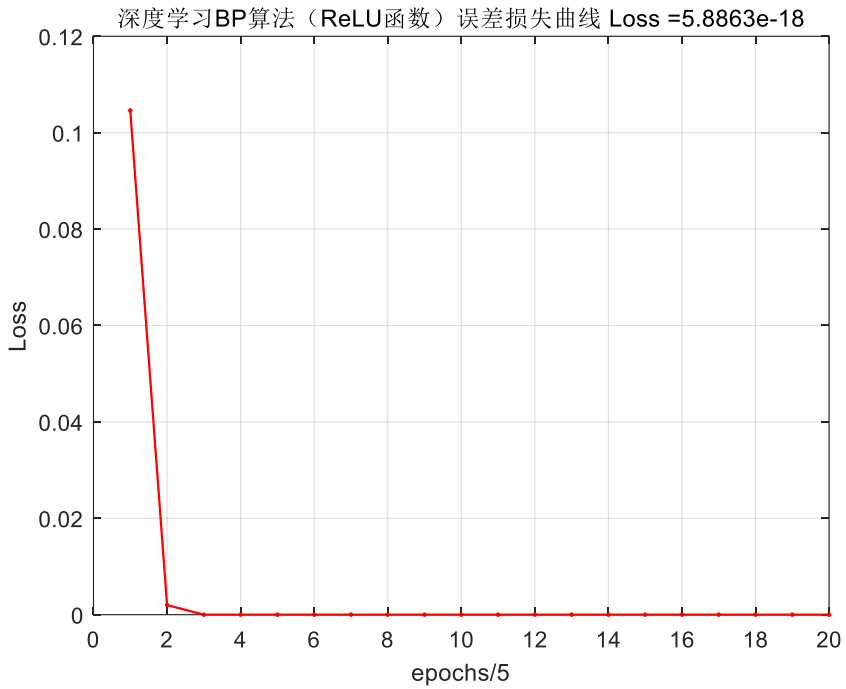
- 单个隐藏层模型最为稳定，重复调用100次，正确率均为100%。
- 但是单隐藏层训练10次不如两个隐藏层训练10次的平均正确率高，均值在75%左右。
- 从添加噪声的训练结果看，单隐藏层精度不错，学习速度也快。

```
>> C = Y(:, 6:10)
C =
    0.9720    0.0000    0.0004    0.0008    0.0008
    0.0036    0.9420    0.0417    0.0001    0.0008
    0.0008    0.0550    0.9516    0.0030    0.0063
    0.0007    0.0027    0.0031    0.8665    0.0001
    0.0030    0.0001    0.0023    0.0079    0.9719
    0.0020    0.0000    0.0002    0.0845    0.0006
    0.0025    0.0000    0.0001    0.0001    0.0023
    0.0001    0.0000    0.0000    0.0012    0.0030
    0.0151    0.0001    0.0007    0.0357    0.0136
    0.0003    0.0000    0.0000    0.0002    0.0006
```

深度学习算法MATLAB实现—ReLU函数

```
Images = loadMNISTImages('MNIST\t10k-images.idx3-ubyte');
Images = reshape(Images, 28, 28, []);
Labels = loadMNISTLabels('MNIST\t10k-labels.idx1-ubyte');
Labels(Labels == 0) = 10;  %类别为0的是10
% 由于图片过多，且训练时间较长，这里选择前200幅图像
X = Images(:,:,1:200);
labels = Labels(1:200);
```

```
>> [Wh, Wo, Y] = DeepLearning_ReLU(X, labels, [1000], 0.01, 100);
第【10】次训练.....
第【20】次训练.....
第【30】次训练.....
第【40】次训练.....
第【50】次训练.....
第【60】次训练.....
第【70】次训练.....
第【80】次训练.....
第【90】次训练.....
第【100】次训练.....
识别正确率为：100.000。
```



使用单隐藏层训练网络，训练200幅图片，训练次数100次，多次调用训练网络，识别正确率为100%，网络比较稳定，未出现梯度消失问题。

深度学习算法MATLAB实现—丢弃节点

```
68      %% 信号前向传播
69      function [v,y] = forwardPropagation(x,Wh,Wo)
70 —        v = cell(lenh,1);
71 —        y = cell(lenh+1,1);
72 —        v{1} = Wh{1}*x;  %输入层到隐藏层权重和
73 —        y{1} = ReLU(v{1}); %修正线性单元激活函数，隐藏层1输出
74 —        y{1} = y{1}.*Dropout(y{1},0.2); % 丢弃第一个隐含层20%的节点
75 —        for ifp = 2:lenh
76 —            v{ifp} = Wh{ifp}*y{ifp-1};
77 —            y{ifp} = ReLU(v{ifp});
78              % 丢弃第ifp个隐含层20%的节点
79 —            y{ifp} = y{ifp}.*Dropout(y{ifp},0.2);
80 —        end
81          %从隐藏层到输出层,softmax激活函数
82 —        y{lenh+1} = Softmax(Wo*y{lenh});
83 —    end
```

```
114     %% 随机丢弃节点
115     function ym = Dropout(y, ratio)
116     % y是输出向量；ratio是输出向量Dropout的比例
117 —       [m1,n1] = size(y);
118 —       ym = zeros(m1,n1);
119 —       num1 = round(m1*n1*(1-ratio));
120 —       idx = randperm(m1*n1, num1); % ym元素的索引
121 —       ym(idx) = 1 / (1-ratio);
122 —   end
```

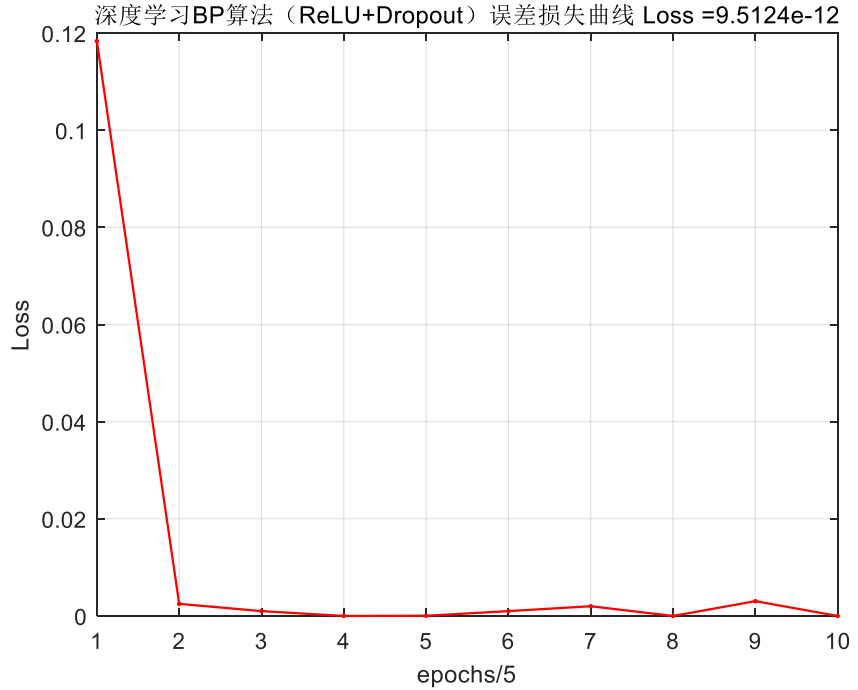
[Wh,Wo,Y] =

DeepLearning_ReLUDropout(X,labels,[800],0.01,50);

第【10】次训练.....

第【50】次训练.....

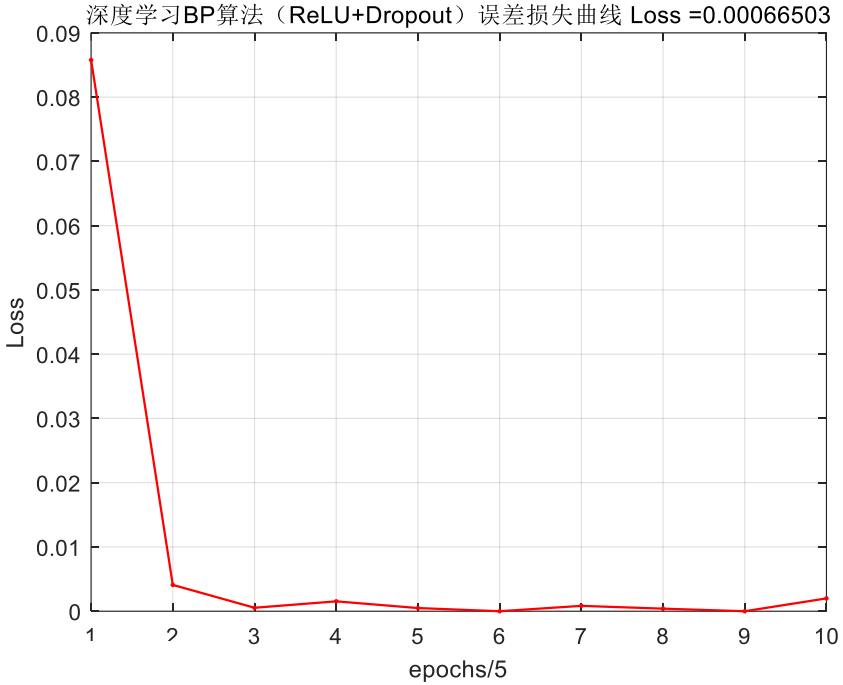
识别正确率为：99.500。%手写数字识别，200幅图像



深度学习算法MATLAB实现—丢弃节点

```
X = Images(:,:,1:500); % 选择前500幅图像
labels = Labels(1:500);
[Wh,Wo,Y] = DeepLearning_ReLUDropout(X,labels,[800],0.01,50);
第【10】次训练.....
第【20】次训练.....
第【30】次训练.....
第【40】次训练.....
第【50】次训练.....
识别正确率为：100.000。
```

```
for i = 1:500
    x = X(:,:,i);
    subplot(10,50,i)
    imshow(x)  %显示500幅手写数字
end
```

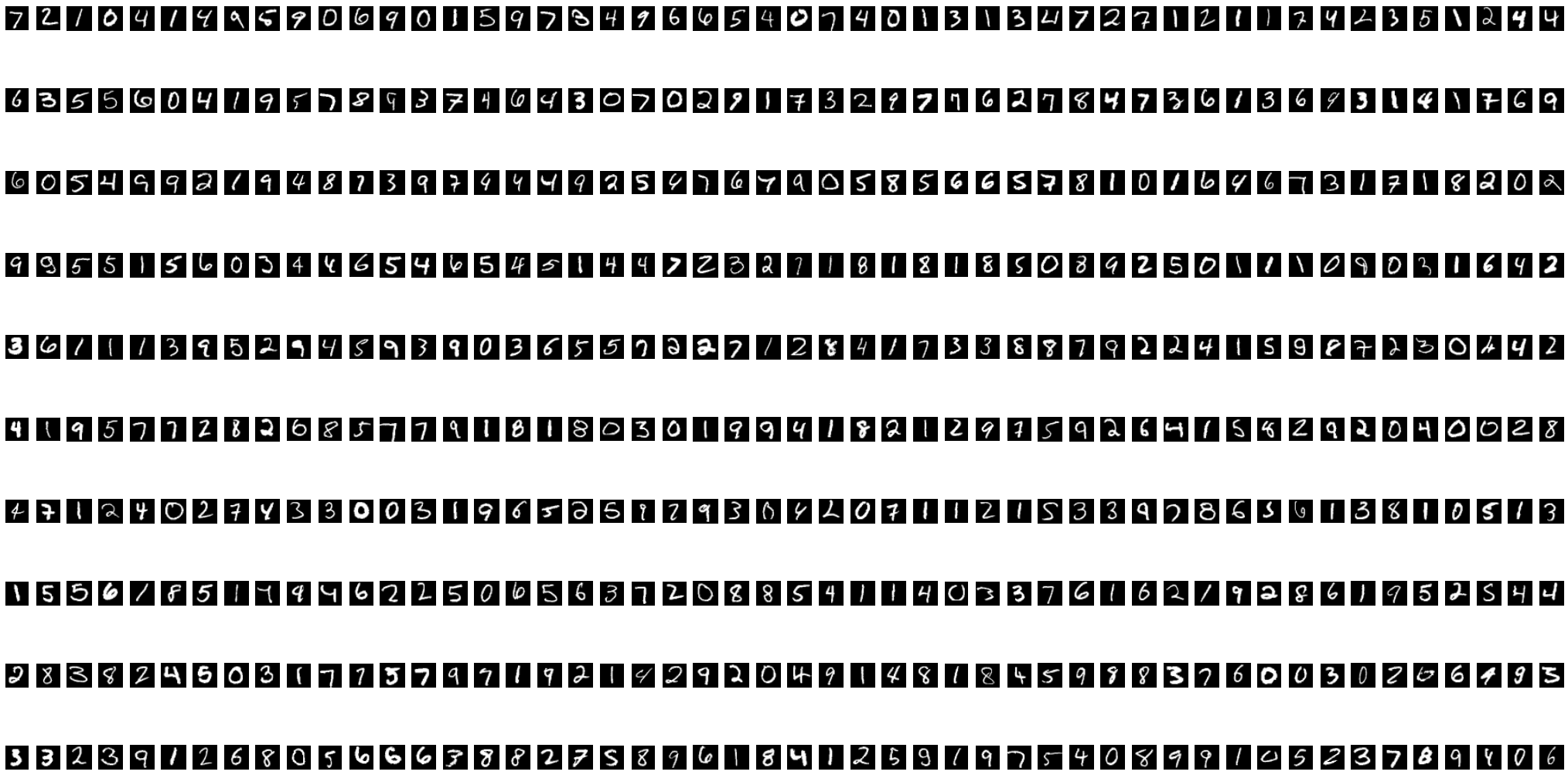



```
>> Class = Y(:,1:10)
Class =
    0.0000    0.0000    1.0000    0.0000    0.0000    1.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    1.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    1.0000    0.0000    1.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    1.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    1.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000    1.0000    0.0000    1.0000
    0.0000    0.0000    0.0000    1.0000    0.0000    0.0000    0.0000    0.0000    0.0000    0.0000
```

从结果来看，经过50次训练，效果不错，识别率100%，且从前十个Y值可以看出，网络训练的精度很高。但网络不太稳定。

深度学习算法MATLAB实现—丢弃节点

信阳师范学院
数学与统计学院
SCHOOL OF MATHEMATICS AND STATISTICS

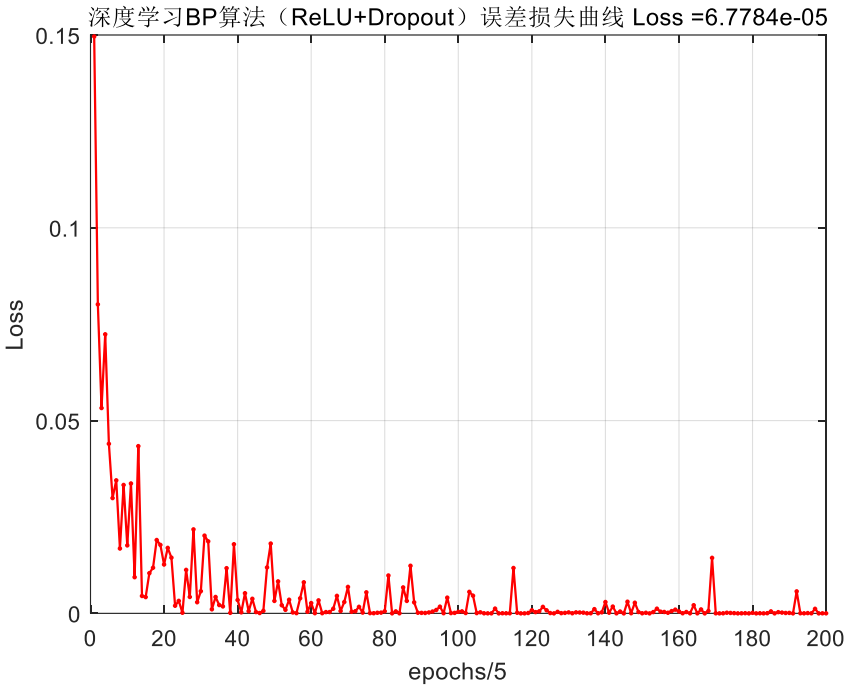


深度学习算法MATLAB实现—丢弃节点

[Wh,Wo,Y] = DeepLearning_ReLUDropout(X,labels,[50],0.01,1000);
% 模型训练1000次，正确率100%，且多次执行均为100%。但是训练100次的结果正确率不高。丢弃节点尽管可以避免过拟合，但是学习速率同时也在降低，需要训练次数更多。

[Wh,Wo,Y] = DeepLearning_ReLUDropout(X,labels,[50,25],0.01,200);
% 两个隐藏层，训练200次，正确率在85%左右徘徊。

[Wh,Wo,Y] =
DeepLearning_ReLUDropout(X,labels,[50,25,10],0.01,200);
% 三个隐藏层，出现了梯度消失问题，且预测Y值常出现NAN值。



单隐藏层，训练1000次误差损失曲线图


01
IDEA
02
LEARNING
03
CREATIVE
INSPIRATION
EDUCATION
BUSINESS
PHOTO BOOK

感谢聆听