深圳大学研究生课程：模式识别理论与方法

课程作业实验报告

实验名称：BP 神经网络

实验编号：Proj06-01

签　　名：

姓　　名：夏荣杰

学　　号：2170269107

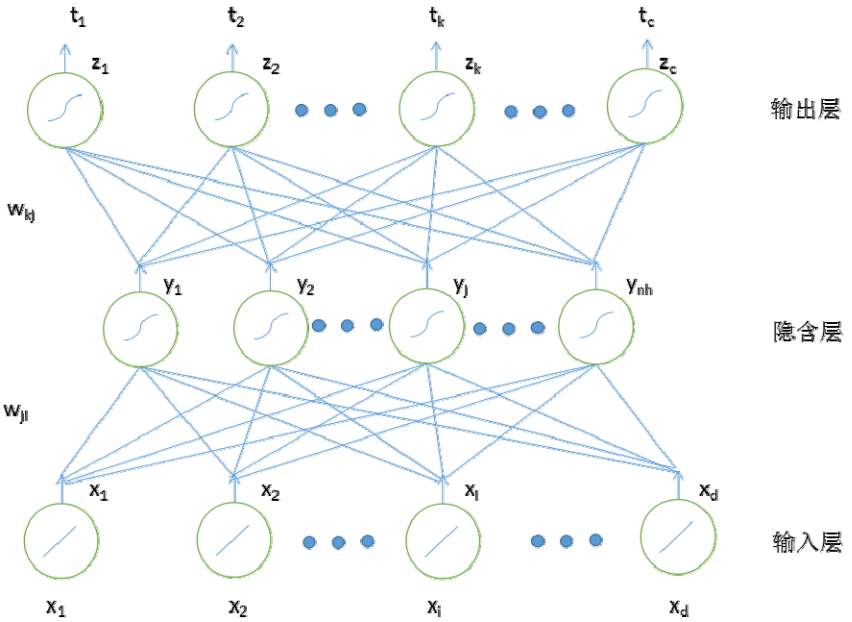
截止提交日期：2018 年 6 月 8 日

摘要：BP(Back Propagation)神经网络是一种按照误差反向传播算法训练的多层前馈神经网络，是目前应用最广泛的神经网络。BP 神经网络的计算过程由前向传播过程和反向传播过程组成。本实验的目的是学习和掌握 BP 神经网络原理、训练算法。本实验设计了 3-3-1 型和 3-4-3 型三层 BP 神经网络，分别对两类和三类样本进行训练。BP 神经网络的设计包括前馈输出、权值更新、训练误差函数以及训练算法学习等几个主要功能的设计。其中最重要的是权值的更新，权值更新采用梯度下降法，根据输出值与教师信号的误差对权值进行反向调整，权值的更新向着减小训练误差的方向调整。在实验过程中，分别采用随机初始化权值和固定初始化权值，并比较这两种初始化权值方式之间的训练误差曲线。实验结果表明，BP 网络的各个权值的初始值和 BP 网络的复杂程度对于 BP 网络的训练学习，包括初始训练误差和最终收敛的训练误差以及训练误差曲线的收敛速度，都具有一定程度上的影响：随机初始化权值和简单的 BP 网络结构的初始训练误差小，最终收敛的训练误差也小；固定初始化权值对应的训练误差曲线更容易收敛。

一、 背景技术 或 基本原理

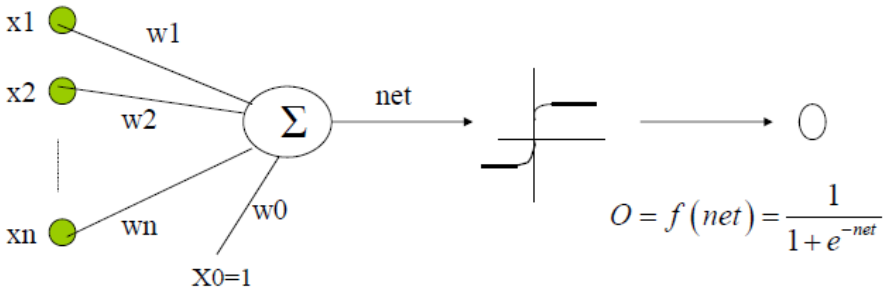
1. 3-层 BP 神经网络

考虑一个 d-n_H-c 的三层 BP 网络，网络有 d 个输入层神经元，有 n_H 个中间层神经元，有 c 个输出层神经元。并规定输入层为线性神经单元。



➢ **BP 网络中的神经元模型**

Sigmoid 函数是 BP 网络中常用的神经元非线性函数：



➢ **Sigmoid 神经元函数及其导数：**

$$f\left(x_1,x_2,...,x_d\right)=f\left(\mathbf{w}^t\mathbf{x}\right)=\frac{1}{1+e^{-\mathbf{w}^t\mathbf{x}}} \qquad (1)$$

$$f'\left(x_1,x_2,...,x_d\right)=f'\left(\mathbf{w}^t\mathbf{x}\right)=f\left(\mathbf{w}^t\mathbf{x}\right)\left(1-f\left(\mathbf{w}^t\mathbf{x}\right)\right) \qquad (2)$$

2. 训练 BP 神经网络：

基本方法：使用梯度下降技术在权空间中寻找使误差函数达到全局最小的点所对应的权值。

定义网络的误差函数： $J\left(w\right)\equiv\frac{1}{2}\sum_{k=1}^{c}\left(t_k-z_k\right)^2=\frac{1}{2}\left\|t-z\right\|^2$

- **BP 网络的信号前向输出：**
  - 神经元函数：

$$f\left(net\right)=\frac{1}{1+e^{-net}} \tag{3}$$

  - 隐含层节点的输出：

$$y_j=f\left(\sum_{i=1}^{d}w_{ji}x_i+w_{j0}\right)\rightarrow\frac{1}{1+e^{-\left(\sum_{i=1}^{d}w_{ji}x_i+w_{j0}\right)}},\ j=1,2,\cdots n_H \tag{4}$$

  - 输出层节点的输出：

$$z_k=f\left(\sum_{j=1}^{n_H}w_{kj}y_j+w_{k0}\right)=f\left(\sum_{j=1}^{n_H}w_{kj}f\left(\sum_{i=1}^{d}w_{ji}x_i+w_{j0}\right)+w_{k0}\right) \tag{5}$$

$$k=1,2,...,c$$

- **BP 网络的误差反向传播：**
  - 利用梯度下降技术：

$$w\left(k+1\right)=w\left(k\right)-\eta\left(k\right)\nabla J\left(w\left(k\right)\right) \tag{6}$$

其中，η 称为学习速率。

权值更新：

$$w\left(m+1\right)=w\left(m\right)+\Delta w\left(m\right),\Delta w=-\eta\frac{\partial J}{\partial w} \tag{7}$$

其中，$\Delta w_{kj}=-\eta\frac{\partial J}{\partial w_{kj}}$，$\Delta w_{ji}=-\eta\frac{\partial J}{\partial w_{ji}}$

  - 计算每一个输出节点 k 的误差：

$$z_k=f\left(net_k\right)=\frac{1}{1+e^{-net_k}},\quad net_k=\sum_{j=1}^{n_H}w_{kj}y_j+w_{k0} \tag{8}$$

$$\begin{aligned}\frac{\partial J}{\partial w_{kj}}&=\frac{\partial J}{\partial z_k}\cdot\frac{\partial z_k}{\partial net_k}\cdot\frac{\partial net_k}{\partial w_{kj}}\\&=-\left(t_k-z_k\right)\cdot f'\left(net_k\right)\cdot y_j\\&=-\left(t_k-z_k\right)\cdot f\left(net_k\right)\cdot\left(1-f\left(net_k\right)\right)\cdot y_j\end{aligned} \tag{9}$$

$$\Delta w_{kj}=-\eta\frac{\partial J}{\partial w_{kj}}=\eta(t_k-z_k)\cdot f(net_k)(1-f(net_k))\cdot y_j \tag{10}$$

  - 计算每一个隐节点 h 的误差：

$$y_j=f\left(net_j\right)=\frac{1}{1+e^{-net_j}},\quad net_j=\sum_{i=1}^{d}w_{ji}x_i+w_{j0} \tag{11}$$

$$\frac{\partial J}{\partial w_{ji}}=\frac{\partial J}{\partial y_j}\cdot\frac{\partial y_j}{\partial net_j}\cdot\frac{\partial net_j}{\partial w_{ji}}$$
$$=\sum_{k=1}^{c}\left[\frac{\partial J}{\partial z_k}\cdot\frac{\partial z_k}{\partial net_k}\cdot\frac{\partial net_k}{\partial y_j}\right]\cdot\frac{\partial y_j}{\partial net_j}\cdot\frac{\partial net_j}{\partial w_{ji}} \quad (12)$$
$$=\sum_{k=1}^{c}\left[\left(t_k-z_k\right)\cdot f'\left(net_k\right)\cdot w_{kj}\right]\cdot f'\left(net_j\right)\cdot x_i$$

$$\Delta w_{ji}=-\eta\frac{\partial J}{\partial w_{ji}}=\eta\sum_{k=1}^{c}[(t_k-z_k)\cdot f'(net_k)\cdot w_{kj}]\cdot f'(net_j)\cdot x_i \quad (13)$$

- 更新网络的每一个权值：

$$w_{kj}\left(m+1\right)=w_{kj}\left(m\right)+\Delta w_{kj}\left(m\right)$$
$$k=1,2,...,c;\ j=1,2,...,n_H \quad (14)$$

$$w_{ji}\left(m+1\right)=w_{ji}\left(m\right)+\Delta w_{ji}\left(m\right)$$
$$j=1,2,...,n_H;i=1,2,...,d \quad (15)$$

3. 基本的 BP 学习算法

- **样本集：** $S=\left\{\left(X_1,t_1\right),\left(X_2,t_2\right),...,\left(X_S,t_S\right)\right\}$
- **基本算法：**
  - 依次使用样本集中的样本(X_k, t_k) 计算出实际输出 Z_k，计算与标准输出的误差 E1；
  - 对 W(1)，W(2)，...，W(L)各做一次调整；
  - 重复这个循环，直到$\sum E_p<\varepsilon$。
- **学习算法的直观解释：误差反传**
  - 用输出层的误差调整输出层权矩阵，并用此误差估计输出层的直接隐含层的误差；
  - 再用隐含层误差估计更前一层的误差，如此获得所有其它各层的误差估计；
  - 用各层的这些估计误差分别对各层权矩阵进行修改。

二、 实验方法 或 算法流程步骤

实验用到的方法和步骤如下：

1. 构造一个 3-3-1 型的三层 BP 神经网络

(1) 创建神经网络结构

输入层单元数为样本的特征维数，输出单元数为目标数量（如类别），隐含层一般来说为一层，若为多层时，每层单元数默认相同，一般单元数大于输入层单元数。

本实验构造一个输入层单元数为 3，隐含层单元数为 3，输出层单元数为 1 的 3-3-1 型的三层 BP 神经网络。

(2) 初始化网络权值

将网络权值的随机初始化一个很小的值，不能全置为 0，全置为 0 后每次更新权重

都会是一样的结果。

本实验中采用两种初始化方式进行重复实验，如下：

随机初始化：在 $-1 \le w \le +1$ 范围内随机初始化所有权值；

固定值初始化：初始化所有权值为 w = 0.5。

(3) 将训练样本循环输入模型中进行训练

根据式（4）—（5），前向传播计算每个神经单元的隐含层节点和输出层节点的输出 Y1 和 Z1；

根据式（9）—（10）计算输出层每一个输出节点的误差 $\Delta w_{kj}\left(m\right)$；

根据式（12）—（13）计算隐含层每一个隐节点的误差 $\Delta w_{ji}\left(m\right)$。

(4) 随机反向传播，更新每个神经单元的权值

① 初始化：隐含层单元数量 n_H，权向量 **w**，准则θ，η，令 $m = 0$

② do $m = m + 1$

③ 随机选择模式 $\mathbf{x}^m$

④ $w_{kj}\left(m+1\right) = w_{kj}\left(m\right) + \Delta w_{kj}\left(m\right)$

⑤ $w_{ji}\left(m+1\right) = w_{ji}\left(m\right) + \Delta w_{ji}\left(m\right)$

⑥ until $\left\|\nabla J\left(\mathbf{w}\right)\right\| < \theta$

⑦ return **w**

⑧ end

用 ω_1 和 ω_2 类的数据对 BP 神经网络进行训练，学习率 $\eta = 0.1$，在 $-1 \le w \le +1$ 范围内随机初始化所有权值。用 t = 0、1 分别表示两个类教师信号。绘出训练误差对训练回合数变化的学习曲线。

重复实验：用 ω_1 和 ω_2 类的数据对 BP 神经网络进行训练，学习率 $\eta = 0.1$，初始化所有权值为 w = 0.5。用 t = 0、1 分别表示两个类教师信号。绘出训练误差对训练回合数变化的学习曲线。

2. 构造一个 3-4-3 型的三层 BP 神经网络

(1) 创建神经网络结构

本实验构造一个输入层单元数为 3，隐含层单元数为 4，输出层单元数为 3 的 3-4-3 型的三层 BP 神经网络。

(2) 初始化网络权值

本实验中采用同实验 1 的两种初始化方式进行重复实验，如下：

随机初始化：在 $-1 \le w \le +1$ 范围内随机初始化所有权值；

固定值初始化：初始化所有权值为 w = 0.5。

(3) 将训练样本循环输入模型中进行训练

根据式（4）—（5），前向传播计算每个神经单元的隐含层节点和输出层节点的输出 Y1 和 Z1；

根据式（9）—（10）计算输出层每一个输出节点的误差 $\Delta w_{kj}\left(m\right)$；

根据式（12）—（13）计算隐含层每一个隐节点的误差 $\Delta w_{ji}\left(m\right)$。

(4) 随机反向传播，更新每个神经单元的权值

① 初始化：隐含层单元数量 n_H，权向量 **w**，准则θ，η，令 $m = 0$

② do $m = m + 1$

③ 随机选择模式 $\mathbf{x}^m$

④ $w_{kj}(m+1) = w_{kj}(m) + \Delta w_{kj}(m)$

⑤ $w_{ji}(m+1) = w_{ji}(m) + \Delta w_{ji}(m)$

⑥ until $\|\nabla J(\mathbf{w})\| < \theta$

⑦ return **w**

⑧ end

用 ω_1、ω_2 和 ω_3 类的数据对 BP 神经网络进行训练，学习率 $\eta = 0.1$，在 $-1 \le w \le +1$ 范围内随机初始化所有权值。用 $t_1 = (1, 0, 0)$，$t_2 = (0, 1, 0)$，$t_3 = (0, 0, 1)$分别表示三个类输出层的教师信号。绘出训练误差对训练回合数变化的学习曲线。

重复实验：用 ω_1、ω_2 和 ω_3 类的数据对 BP 神经网络进行训练，学习率 $\eta = 0.1$，初始化所有权值为 w = 0.5。用 $t_1 = (1, 0, 0)$，$t_2 = (0, 1, 0)$，$t_3 = (0, 0, 1)$分别表示三个类输出层的教师信号。绘出训练误差对训练回合数变化的学习曲线。

三、 实验结果

1. 构造一个 3-3-1 型的三层 BP 神经网络

用 ω_1 和 ω_2 类的数据对 3-3-1 型的三层 BP 神经网络进行训练，学习率 $\eta = 0.1$。

- **不同的权值初始化方法下，训练误差对训练回合数变化的学习曲线对比**

在 $-1 \le w \le +1$ 范围内随机初始化所有权值，初始化所有权值为 w = 0.5 这两种不同的权值初始化方法，训练误差对训练回合数变化的学习曲线对比如图 1 所示。

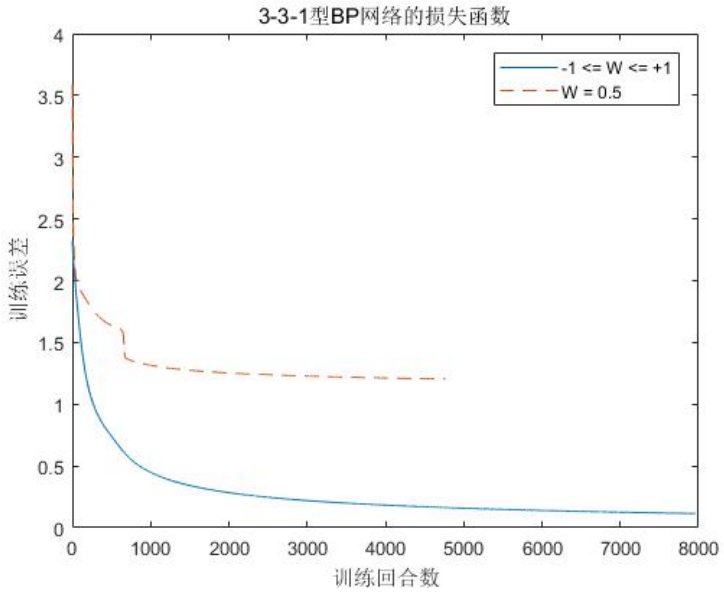


图 1. 在 $-1 \le w \le +1$ 范围内随机初始化所有权值和初始化所有权值为 w = 0.5，训练误差对训练回合数变化的学习曲线对比

➢ **不同的权值初始化方法下，损失函数收敛时所需要的训练回合数对比**

在 $-1 \le w \le +1$ 范围内随机初始化所有权值和初始化所有权值为 w = 0.5，训练误差对训练回合数变化的学习曲线收敛时所需要的训练回合数对比如下：

随机初始化权值的训练回合为：7960
固定初始化权值的训练回合为：4763

2. 构造一个 3-4-3 型的三层 BP 神经网络

用 ω_1、ω_2 和 ω_3 类的数据对 3-4-3 型的三层 BP 神经网络进行训练，学习率 $\eta = 0.1$。

➢ **不同的权值初始化方法下，训练误差对训练回合数变化的学习曲线对比**

在 $-1 \le w \le +1$ 范围内随机初始化所有权值，初始化所有权值为 w = 0.5 这两种不同的权值初始化方法，训练误差对训练回合数变化的学习曲线对比如图 2 所示。

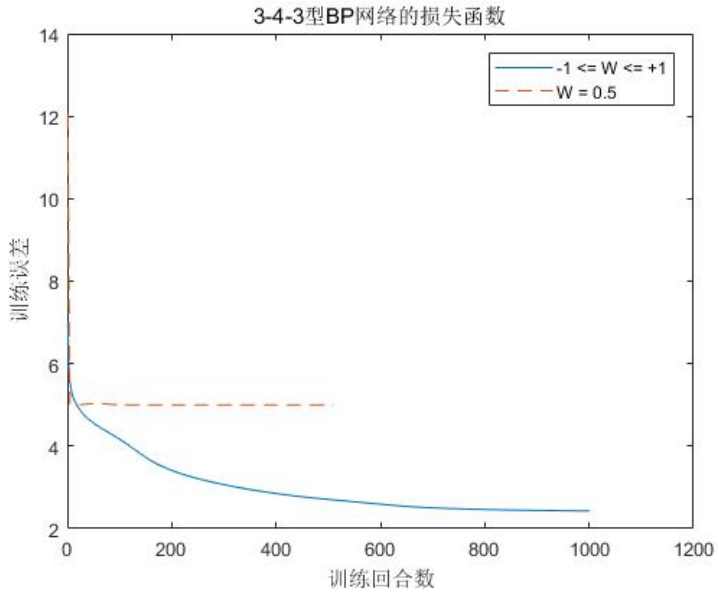


图 2. 在 $-1 \le w \le +1$ 范围内随机初始化所有权值，初始化所有权值为 w = 0.5，训练误差对训练回合数变化的学习曲线对比

➢ **不同的权值初始化方法下，损失函数收敛时所需要的训练回合数对比**

在 $-1 \le w \le +1$ 范围内随机初始化所有权值和初始化所有权值为 w = 0.5，训练误差对训练回合数变化的学习曲线收敛时所需要的训练回合数对比如下：

随机初始化权值的训练回合为：1000
固定初始化权值的训练回合为：509

四、 讨论与分析

1. 构造一个 3-3-1 型的三层 BP 神经网络

如图 1 所示，实线是在 $-1 \le w \le +1$ 范围内随机初始化所有权值，ω_1 和 ω_2 类的训练误差对训练回合数变化的学习曲线。虚线是初始化所有权值为 w = 0.5，ω_1 和 ω_2 类的训练误差对训练回合数变化的学习曲线。

由图 1 可以看出，无论是随机初始化权值还是固定初始化权值，训练误差随着训练回合数的增加不断减小，经过不断地迭代训练，训练误差趋于平稳，接近于水平直线。相比固定初始化权值为 w = 0.5，随机初始化权值的曲线（实线）初始训练误差较低（如下图 3 所示）；当网络达到稳定时的训练误差大大减少。

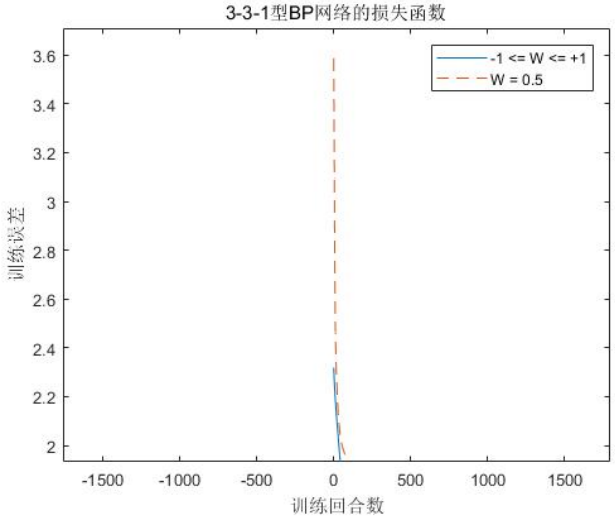


图 3. 在 $-1 \le w \le +1$ 范围内随机初始化所有权值，初始化所有权值为 w = 0.5，初始训练误差对比
（图 3 是由图 1 在训练回合数为 0 处经过放大所得）

因此，权值的初始值的选取对神经网络的训练有一定的影响，而如何选取也是一个值得研究的问题。

对比上面实验结果中训练误差对训练回合数变化的学习曲线收敛时所需要的训练回合数：随机初始化权值的训练回合数为 7960，固定初始化权值的训练回合数为 4763，可见，固定初始化权值的曲线更容易收敛。

2. 构造一个 3-4-3 型的三层 BP 神经网络

如图 2 所示，实线是在 $-1 \le w \le +1$ 范围内随机初始化所有权值，ω_1、ω_2 和 ω_3 类的训练误差对训练回合数变化的学习曲线。虚线是初始化所有权值为 w = 0.5，ω_1、ω_2 和 ω_3 类的训练误差对训练回合数变化的学习曲线。

由图 2 可以得出同实验 1 的结论：无论是随机初始化权值还是固定初始化权值，训练误差随着训练回合数的增加不断减小，经过不断地迭代训练，训练误差趋于平稳，接近于水平直线。相比固定初始化权值为 w = 0.5，随机初始化权值的曲线（实线）初始训练误差较低（如下图 4 所示）；此外，当网络达到稳定时的训练误差大大减少。

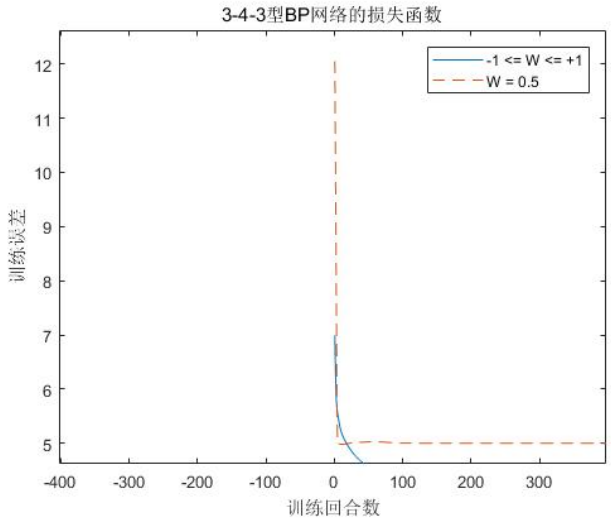


图 4. 在 $-1 \le w \le +1$ 范围内随机初始化所有权值，初始化所有权值为 w = 0.5，初始训练误差对比

（图 4 是由图 2 在训练回合数为 0 处经过放大所得）

对比上面实验结果中训练误差对训练回合数变化的学习曲线收敛时所需要的训练回合数：随机初始化权值的训练回合数为 1000，固定初始化权值的训练回合数为 509，可见，固定初始化权值的曲线更容易收敛。

3. 3-3-1 型和 3-4-3 型的三层 BP 神经网络对比：

- **二者之间的共同点：**

对于 3-3-1 型和 3-4-3 型的三层 BP 神经网络，无论是随机初始化权值还是固定初始化权值，它们的训练误差随着训练回合数的增加不断减小，经过不断地迭代训练，训练误差趋于平稳。相比固定初始化权值，随机初始化权值的曲线的初始训练误差较低；当网络达到稳定时的训练误差大大减少。此外，固定初始化权值的曲线更容易收敛。

- **二者之间的不同点：**

对比图 3 和图 4，可以看出，无论是随机初始化权值还是固定初始化权值，图 4 的初始训练误差比图 3 的都高，即 3-4-3 型的三层 BP 神经网络的初始训练误差比 3-3-1 型的三层 BP 神经网络的要高。

对比图 1 和图 2，可以看出，无论是随机初始化权值还是固定初始化权值，图 2 最终的训练误差比图 1 的都高，即 3-4-3 型的三层 BP 神经网络的训练误差比 3-3-1 型的三层 BP 神经网络的要高。

这是因为每一层的神经单元越多，所构造的网络越复杂，需要大量的训练样本才能达到预期的目标。可以尝试的增加训练次数，或者提高训练样本的数量。

总结：BP 网络的各个权值的初始值和 BP 网络的复杂程度对于 BP 网络的训练学习，包括初始训练误差和最终收敛的训练误差以及训练误差曲线的收敛速度，都具有一定程度上的影响：随机初始化权值和简单的 BP 网络结构的初始训练误差小，最终收敛的训练误差也小；固定初始化权值对应的训练误差曲线更容易收敛。

附录.

```
%%------------------Proj06-01：BP 神经网络--------------------%%
clear; clc;
%%第一类
w1 = [1.58 2.32 -5.8; 0.67 1.58 -4.78; 1.04 1.01 -3.63; -1.49 2.18 -0.39; -0.41
1.21 -4.73;
    1.39 3.16 2.87; 1.20 1.40 -1.89; -0.92 1.44 -3.22; 0.45 1.33 -4.38; -0.76 0.84
-1.96];
%%第二类
w2 = [0.21 0.03 -2.21; 0.37 0.28 -1.8; 0.18 1.22 0.16; -0.24 0.93 -1.01; -1.18
0.39 -0.39;
    0.74 0.96 -1.16; -0.38 1.94 -0.48; 0.02 0.72 -0.17; 0.44 1.31 -0.14; 0.46 1.49
0.68];
%%第三类
w3 = [-1.54 1.17 0.64; 5.41 3.45 -1.33; 1.55 0.99 2.69; 1.86 3.19 1.51; 1.68 1.79
-0.87;
    3.51 -0.22 -1.39; 1.40 -0.44 0.92; 0.44 0.83 1.97; 0.25 0.68 -0.99; -0.66 -0.45
0.08];
```

1. 构造一个 3-3-1 型的三层 BP 神经网络

```
%% 构造一个 3-3-1 型的三层 BP 神经网络
X = [w1; w2];%BP 网络输入层
t1 = 1; t2 = 0;%教师信号
T = [repmat(t1, 1, size(w1, 1)), repmat(t2, 1, size(w2, 1))];%输出层对应的样本点
数个教师信号

eta = 0.1;%学习率
theta = 0.001;
figure;
for selection_rand = [1, 0]%%初始化各权值和偏置
    if selection_rand%随机初始化
        Wji = -1 + 2 * rand(3, 3);%输入层到中间层的权值矩阵
        Wjb = -1 + 2 * rand(3, 1);%中间层神经元的偏置向量
        Wkj = -1 + 2 * rand(1, 3);%中间层到输出层的权值矩阵
        Wkb = -1 + 2 * rand(1, 1);%输出层神经元的偏置向量
    else%初始化为固定值
        Wji = 0.5 * ones(3, 3);%输入层到中间层的权值矩阵
        Wjb = 0.5 * ones(3, 1);%中间层神经元的偏置向量
        Wkj = 0.5 * ones(1, 3);%中间层到输出层的权值矩阵
        Wkb = 0.5 * ones(1, 1);%输出层神经元的偏置向量
    end
    m = 0;
    J = [];
```

```
    while(1)
        m = m + 1;
        [Wji, Wjb, Wkj, Wkb, d_wkj, d_wji, Z1] = BP_train(eta, Wji, Wjb, Wkj, Wkb,
X, T);
        J = [J, 1/2 * (norm(T - Z1))^2];
        if norm(d_wji) < theta && norm(d_wkj) < theta || m > 10000
            break;
        end
    end
    if selection_rand
        plot(J, '-'); fprintf('随机初始化权值的训练回合为：%d\n', m - 1);
    else
        plot(J, '--'); fprintf('固定初始化权值的训练回合为：%d\n', m - 1);
    end
    hold on;
end
title('3-3-1 型 BP 网络的损失函数'); legend('-1 <= W <= +1', 'W = 0.5');
xlabel('训练回合数'); ylabel('训练误差');
```

2. 构造一个 3-4-3 型的三层 BP 神经网络

```
%% 构造一个 3-4-3 型的三层 BP 神经网络
X = [w1; w2; w3];%BP 网络输入层
t1 = [1 0 0]; t2 = [0 1 0]; t3 = [0 0 1];%教师信号
T = [repmat(t1', 1, size(w1, 1)), repmat(t2', 1, size(w2, 1)), repmat(t3', 1,
size(w3, 1))];%输出层对应的样本点数个教师信号

eta = 0.1;%学习率
theta = 0.0001;
figure;
for selection_rand = [1, 0]%%初始化各权值和偏置
    if selection_rand%随机初始化
        Wji = -1 + 2 * rand(4, 3);%输入层到中间层的权值矩阵
        Wjb = -1 + 2 * rand(4, 1);%中间层神经元的偏置向量
        Wkj = -1 + 2 * rand(3, 4);%中间层到输出层的权值矩阵
        Wkb = -1 + 2 * rand(3, 1);%输出层神经元的偏置向量
    else%初始化为固定值
        Wji = 0.5 * ones(4, 3);%输入层到中间层的权值矩阵
        Wjb = 0.5 * ones(4, 1);%中间层神经元的偏置向量
        Wkj = 0.5 * ones(3, 4);%中间层到输出层的权值矩阵
        Wkb = 0.5 * ones(3, 1);%输出层神经元的偏置向量
    end
    m = 0;
    J = [];
    while(1)
```

```
        m = m + 1;
        [Wji, Wjb, Wkj, Wkb, d_wkj, d_wji, Z1] = BP_train(eta, Wji, Wjb, Wkj, Wkb,
X, T);
        J = [J, 1/2 * (norm(T - Z1))^2];
        if norm(d_wji) < theta && norm(d_wkj) < theta || m > 1000
            break;
        end
    end
    if selection_rand
        plot(J, '-'); fprintf('随机初始化权值的训练回合为：%d\n', m - 1);
    else
        plot(J, '--'); fprintf('固定初始化权值的训练回合为：%d\n', m - 1);
    end
    hold on;
end
title('3-4-3 型 BP 网络的损失函数'); legend('-1 <= W <= +1', 'W = 0.5');
xlabel('训练回合数'); ylabel('训练误差');
```

3. BP 神经网络学习算法

```
function [Wji, Wjb, Wkj, Wkb, d_wkj, d_wji, Z1] = BP_train(eta, Wji, Wjb, Wkj,
Wkb, X, T)
    %%该子函数用于 BP 网络学习
    %输入：学习率 eta，初始权值 Wji, Wjb, Wkj, Wkb，输入层输入向量 X，输出层教师信号 T
    %输出：更新权值 Wji, Wjb, Wkj, Wkb，权值变化量 d_wkj, d_wji，输出层输出向量 Z1
    %%前馈输出
    Y0 = Wji * X' + Wjb;
    Y1 = 1 ./ (1 + exp(-Y0));
    Z0 = Wkj * Y1 + Wkb;
    Z1 = 1 ./ (1 + exp(-Z0));
    %计算梯度
    dZ = (T - Z1) .* (Z1 .* (1 - Z1));
    d_wkj = eta * dZ * Y1';
    d_wji = eta * (Wkj' * dZ .* Y1 .* (1 - Y1)) * X;
    %更新权值
    Wkj = Wkj + d_wkj;
    Wji = Wji + d_wji;
end
```