Homework 2

Shen Jiyuan   5130309194

==

一、推导题中 MLQP 的两种 BP 算法：

b) 在线学习

1、已知关系式罗列：

① $e_j(n)=d_j(n)-y_j(n)$，神经元 j 第 n 次迭代的误差

② $\varepsilon(n)=\frac{1}{2}\sum_{j\in C}e_j^2(n)$，神经网络第 n 次迭代的误差能量

③ $\gamma_j(n)=\sum_{i=0}^{m}(u_{ji}(n)y_i^2(n)+v_{ji}(n)y_i(n))$，神经元 j 第 n 次迭代时 φ_j 输入

④ $y_j(n)=\varphi_j(\gamma_j(n))$，神经元 j 第 n 次迭代后输出

2、计算偏微分：

① $\frac{\partial\varepsilon(n)}{\partial e_j(n)}=e_j(n)$　② $\frac{\partial e_j(n)}{\partial y_j(n)}=-1$　③ $\frac{\partial y_j(n)}{\partial\gamma_j(n)}=\varphi_j'(\gamma_j(n))$

④ $\frac{\partial\gamma_j(n)}{\partial u_{ji}(n)}=y_i^2(n)$　⑤ $\frac{\partial\gamma_j(n)}{\partial v_{ji}(n)}=y_i(n)$

3、依照如下链式求导法则：

$$\frac{\partial\varepsilon(n)}{\partial u_{ji}(n)}=\frac{\partial\varepsilon(n)}{\partial e_j(n)}\cdot\frac{\partial e_j(n)}{\partial y_j(n)}\cdot\frac{\partial y_j(n)}{\partial\gamma_j(n)}\cdot\frac{\partial\gamma_j(n)}{\partial u_{ji}(n)}$$

$$\frac{\partial\varepsilon(n)}{\partial v_{ji}(n)}=\frac{\partial\varepsilon(n)}{\partial e_j(n)}\cdot\frac{\partial e_j(n)}{\partial y_j(n)}\cdot\frac{\partial y_j(n)}{\partial\gamma_j(n)}\cdot\frac{\partial\gamma_j(n)}{\partial v_{ji}(n)}$$

得到权值向量修正量：

$$\Delta u_{ji}(n)=-\eta\cdot\frac{\partial\varepsilon(n)}{\partial u_{ji}(n)}=\eta\cdot e_j(n)\cdot\varphi_j'(\gamma_j(n))\cdot y_i^2(n)=\eta\cdot\delta_j(n)\cdot y_i^2(n)$$

$$\Delta v_{ji}(n)=-\eta\cdot\frac{\partial\varepsilon(n)}{\partial v_{ji}(n)}=\eta\cdot e_j(n)\cdot\varphi_j'(\gamma_j(n))\cdot y_i(n)=\eta\cdot\delta_j(n)\cdot y_i(n)$$

4、推导修正量中的 $\delta_j(n)$：

⑴ $\delta_j(n)=e_j(n)\cdot\varphi_j'(\gamma_j(n))=\frac{\partial\varepsilon(n)}{\partial\gamma_j(n)}=-\frac{\partial\varepsilon(n)}{\partial y_j(n)}\cdot\frac{\partial y_j(n)}{\partial\gamma_j(n)}=-\frac{\partial\varepsilon(n)}{\partial y_j(n)}\cdot\varphi_j'(\gamma_j(n))$

⑵ 由已知关系式：

① $\varepsilon(n)=\frac{1}{2}\sum_{k\in C}e_k^2(n)$　② $e_k(n)=d_k(n)-y_k(n)=d_k(n)-\varphi_k(\gamma_k(n))$

③ $\gamma_k(n)=\sum_{j=0}^{m}(u_{kj}(n)y_j^2(n)+v_{kj}(n)y_j(n))$

推导得到：

① $\frac{\partial \varepsilon(n)}{\partial y_j(n)} = \sum_{k\in C} e_k(n)\cdot\frac{\partial e_k(n)}{\partial y_j(n)} = \sum_{k\in C} e_k(n)\cdot\frac{\partial e_k(n)}{\partial \gamma_k(n)}\cdot\frac{\partial \gamma_k(n)}{\partial y_j(n)}$

② $\frac{\partial e_k(n)}{\partial \gamma_k(n)} = -\varphi_k^{'}(\gamma_k(n))$

③ $\frac{\partial \gamma_k(n)}{\partial y_j(n)} = 2u_{kj}(n)y_j(n) + v_{kj}(n)$

④ $\frac{\partial \varepsilon(n)}{\partial y_j(n)} = -\sum_{k\in C} e_k(n)\cdot\varphi_k^{'}(\gamma_k(n))\cdot\frac{\partial \gamma_k(n)}{\partial y_j(n)} = -\sum_{k\in C}\delta_k(n)\cdot(2u_{kj}(n)y_j(n)+v_{kj}(n))$

⑶ 得到修正量中的 $\delta_j(n)$ 为

$$\delta_j(n) = \varphi_j^{'}(\gamma_j(n))\cdot\sum_{k\in C}\delta_k(n)\cdot(2u_{kj}(n)y_j(n)+v_{kj}(n))$$

5、故最终的权值修正公式为：

$$\Delta u_{ji}(n) = \alpha\Delta u_{ji}(n-1) + \eta\cdot\delta_j(n)\cdot y_i^2(n)$$

$$\Delta v_{ji}(n) = \alpha\Delta v_{ji}(n-1) + \eta\cdot\delta_j(n)\cdot y_i(n)$$

a) 批量学习

1、与在线学习相比，批量学习的代价计算函数为：

$$\varepsilon_{av} = \frac{1}{2N}\sum_{n=1}^{N}\sum_{j\in C} e_j^2(n) = \frac{1}{N}\sum_{n=1}^{N}\varepsilon(n)$$

2、相应的权值修正公式为：

$$\Delta u_{ji} = -\eta\cdot\frac{\partial \varepsilon_{av}}{\partial u_{ji}} = -\frac{\eta}{N}\sum_{n=1}^{N}\frac{\partial \varepsilon(n)}{\partial u_{ji}}$$

$$\Delta v_{ji} = -\eta\cdot\frac{\partial \varepsilon_{av}}{\partial v_{ji}} = -\frac{\eta}{N}\sum_{n=1}^{N}\frac{\partial \varepsilon(n)}{\partial v_{ji}}$$

由前面的推导可以知道：

① $\frac{\partial \varepsilon(n)}{\partial u_{ji}} = -\delta_j(n)y_i^2(n)$ ② $\frac{\partial \varepsilon(n)}{\partial v_{ji}} = -\delta_j(n)y_j(n)$

③ $\delta_j(n) = \varphi_j^{'}(\gamma_j(n))\cdot\sum_{k\in C}\delta_k(n)\cdot(2u_{kj}(n)y_j(n)+v_{kj}(n))$

3、故最终的权值修正公式为：

$$\Delta u_{ji} = -\frac{\eta}{N}\sum_{n=1}^{N}\frac{\partial \varepsilon(n)}{\partial u_{ji}} = \frac{\eta}{N}\sum_{n=1}^{N}\delta_j(n)y_i^2(N)$$

$$\Delta v_{ji} = -\frac{\eta}{N}\sum_{n=1}^{N}\frac{\partial \varepsilon(n)}{\partial v_{ji}} = \frac{\eta}{N}\sum_{n=1}^{N}\delta_j(n)y_i(N)$$

==

==

二、双螺旋问题分解为 16 个两类子问题，分别用两层 BP 算法学习再用 MINMAX 算法求解。

1、设定的网络参数为：

学习速率：0.01

网络：[30, 1] (hidden-output)

2、16 个子问题的分解思想

输入文件’train_all.txt’中有 96 组训练数据，其中训练结果为 1 和训练结果为 0 的各占 48 组。因此，将训练结果为 1 的再分为 4 个问题（每个 12 组训练数据），同理处理训练结果为 0 的训练数据。

即相当于：

▪描述输入文件为(数字为行数)：

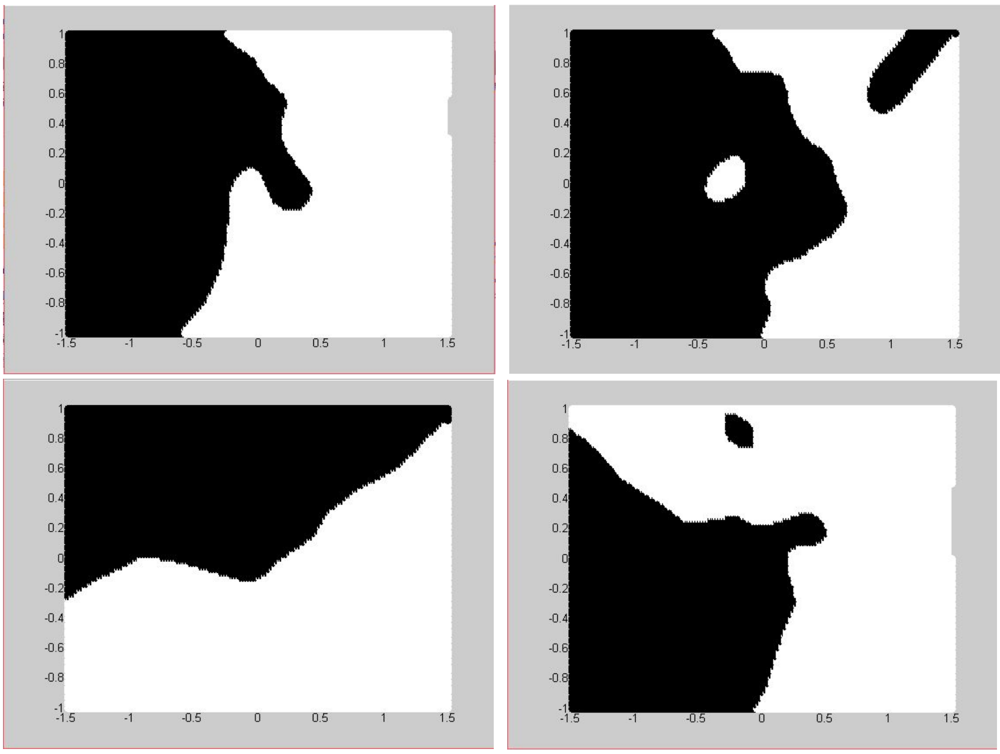
E(1,3,5,7,9,11,13,15,17,19,21,23); A(2,4,6,8,10,12,14,16,18,20,22,24);
F(25,27,29,31,33,35,37,39,41,43,45,47); B(26,28,30,32,34,36,38,40,42,44,46,48);
G(49,51,53,55,57,59,61,63,65,67,69,71);C(50,52,54,56,58,60,62,64,66,68,70,72);
H(73,75,77,79,81,83,85,87,89,91,93,95);D(74,76,78,80,82,84,86,88,90,92,94,96).
(E,F,G,H 为 1；A,B,C,D 为 0)

▪分解的子问题输入为：（文件中 01 相间，提高学习准确度）

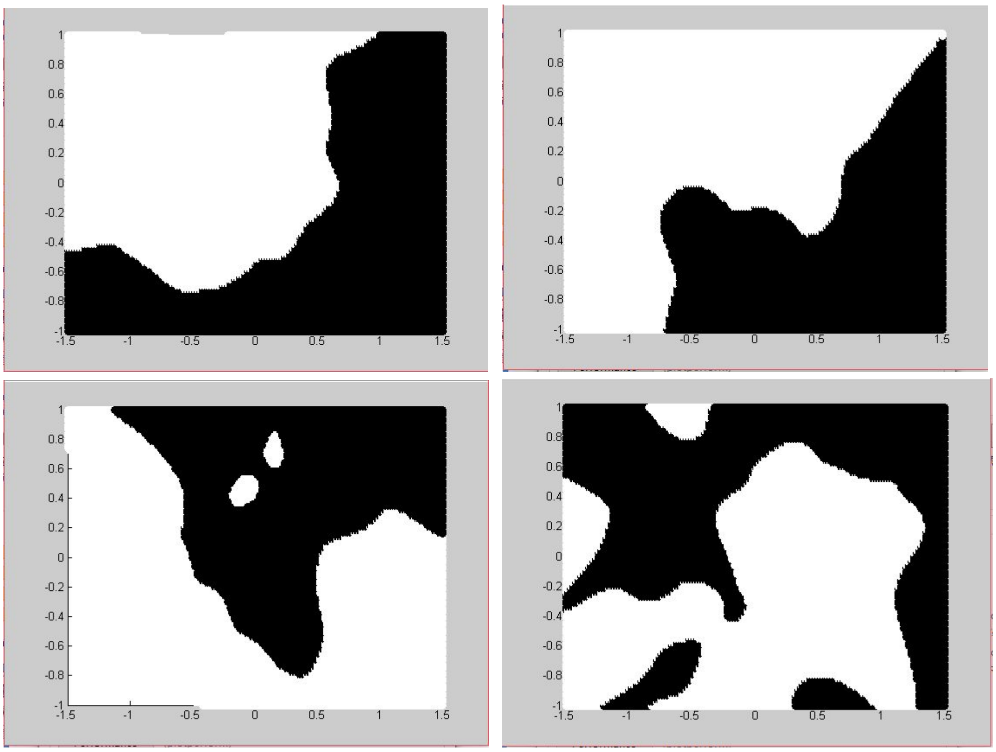
train_1:=EA; train_2:=EB; train_3:=EC; train_4:=ED;
train_5:=FA; train_6:=FB; train_7:=FC; train_8:=FD;
train_9:=GA; train_10:=GB; train_11:=GC; train_12:=GD;
train_13:=HA; train_14:=HB; train_15:=HC; train_16:=HD;

3、16 个子问题的分界面

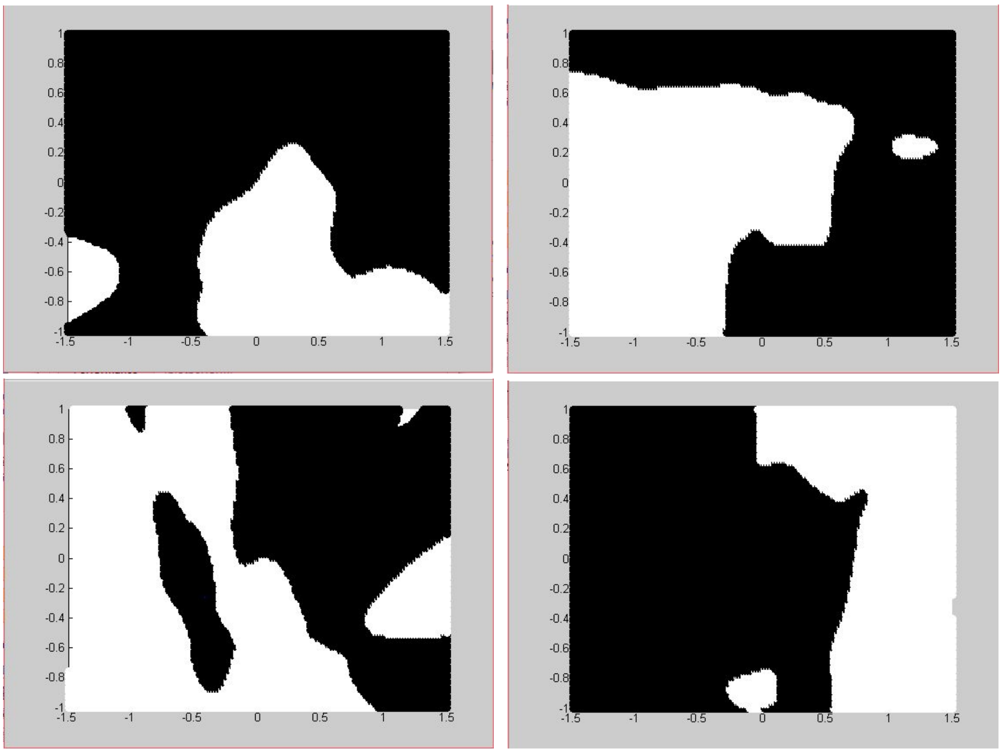
train_1~train_4



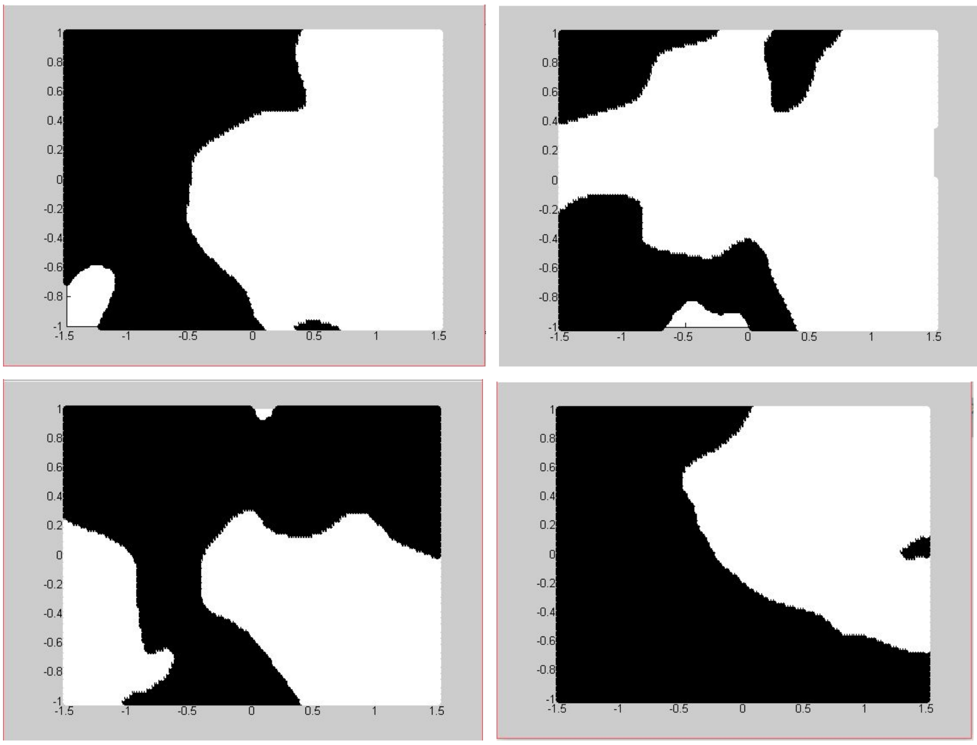
train_5~train_8



train_9~train_12

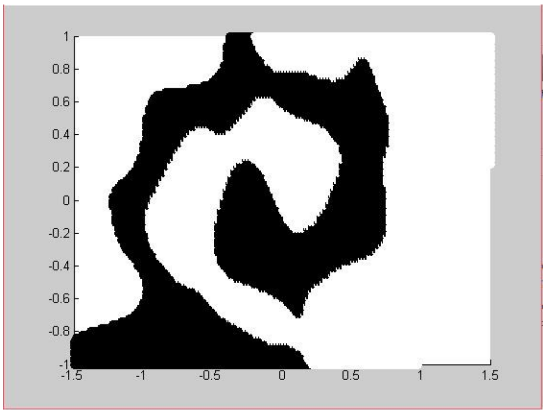


train_13~train_16



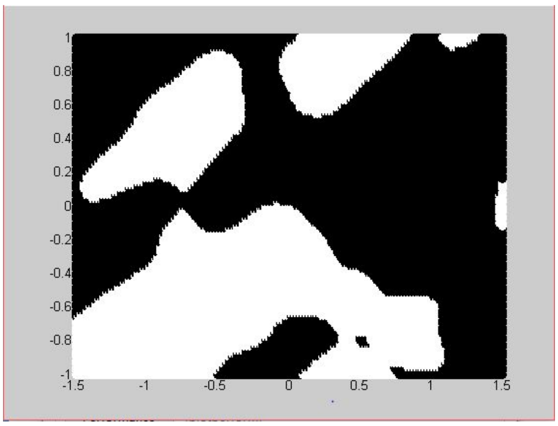
4、16 个子题经过 MINMAX 算法之后形成的双螺旋分界面

每四个矩阵 MIN(or),再将四个 MIN 的矩阵 MAX(and)。矩阵即代码中的 Y 矩阵。



==

三、双螺旋问题直接用两层 BP 算法学习求解。(两层前向网络直接学习得到双螺旋分界面)



==

===

四、比较分析二、三中的训练时间,误差能量和决策面。

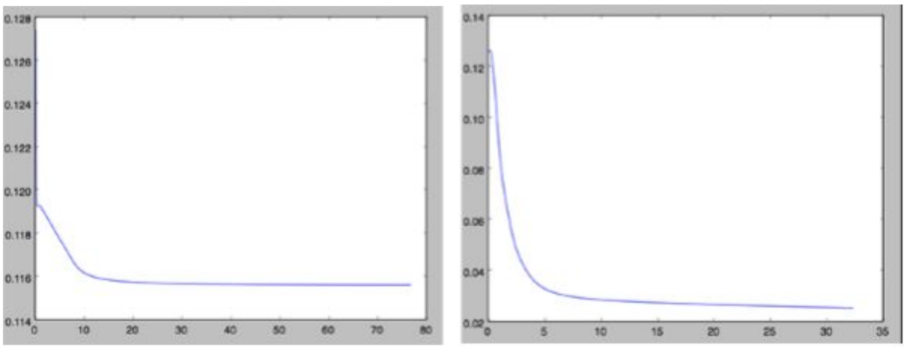
题二是经过 MINMAX 优化的 BP 算法；题三是直接的 BP 算法。

- 训练时间

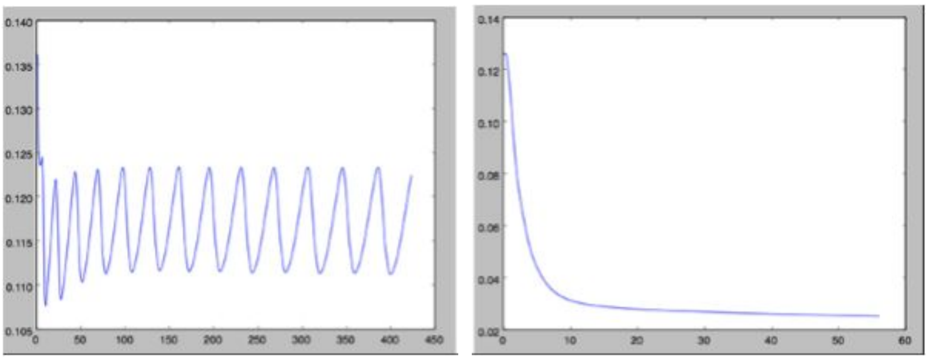
经过 MINMAX 优化的 BP 算法需要处理 384 组训练数据（包括重复），16 个输入以及 MINMAX 计算；相比较而言，直接的 BP 算法需要处理 96 组训练数据以及 1 个输入（无 MINMAX 计算）。因此直接的 BP 算法的训练时间必然较短一些。（即使训练效果和时间并无关系）

- 网络误差能量 BP - MINMAX

（隐形神经元个数为 2）



（隐形神经元个数为 30）



当训练数据规模较大时，BP 算法缓慢且不稳定；MINMAX 则可使误差更快收敛，从而得到更为正确的分类。

- 决策面

直接使用 BP 算法的分界面不理想，而 MINMAX 优化的 BP 算法可以将多数测试数据进行正确分类，但仍有一些特殊的点不能够识别，这与网络的初始值、结构等有关，可以通过调整网络的参数以适当提高网络的性能。对于较复杂的函数而言，可能存在一些局部极小值，一旦输入点靠近这些值，网络将难以为这些点正确分类。

===