第5章 径向基函数神经网络

CHAPTER 5

神经网络根据函数逼近方式可分为全局逼近网络和局部逼近网络，对于全局逼近网络，其任意可调参数的变化都会引起网络所有输出节点的变化，例如，BP 神经网络。而对于局部逼近网络，网络输入空间的某个局部区域只有少数几个连接权值影响网络的输出，例如径向基函数(Radial Basis Function，RBF)神经网络。因此，根据以上网络结构可知，局部逼近网络在学习效率和实时性方面显著优于全局逼近网络。本章将以 RBF 神经网络为例，分别从插值和分类的角度介绍正则化 RBF 神经网络和广义 RBF 神经网络。

视频讲解

5.1 正则化 RBF 神经网络

当用 RBF 神经网络解决非线性映射问题时，通常采用插值的观点来理解，因此本节将对插值问题进行描述，并给出通过径向基函数解决插值问题的方法，由此引出正则化 RBF 神经网络的结构。

5.1.1 插值问题

设 N 维空间有 P 个输入向量 $\boldsymbol{X}^p$，它们在输出空间相应的目标值为 d^p，$p=1,2,\cdots,P$，P 对输入-输出样本构成了训练样本集。插值的目的是寻找一个非线性映射函数 $F(\boldsymbol{X})$，使其满足下述插值条件：

$$F(\boldsymbol{X}^p)=d^p, \quad p=1,2,\cdots,P \tag{5-1}$$

式中，函数 F 表示插值曲面，该插值曲面必须通过所有训练数据点。

5.1.2 径向基函数解决插值问题

采用径向基函数解决插值问题的方法是，选择 P 个基函数，每一个基函数对应一个训练数据，各基函数的形式为

$$\varphi(\|\boldsymbol{X}-\boldsymbol{X}^p\|), \quad p=1,2,\cdots,P \tag{5-2}$$

式中，基函数 φ 为非线性函数，训练数据点 $\boldsymbol{X}^p$ 是 φ 的数据中心。基函数以输入空间的点 $\boldsymbol{X}$ 与中心 $\boldsymbol{X}^p$ 的距离作为函数的自变量。由于距离是径向同性的，故函数 φ 被称为径向基函数。基于径向基函数技术的插值函数定义为基函数的线性组合，即

$$\boldsymbol{F}(\boldsymbol{X})=\sum_{p=1}^{P}\omega_p\varphi(\|\boldsymbol{X}-\boldsymbol{X}^p\|) \tag{5-3}$$

由于所有的数据点均位于插值函数上，因此将式(5-1)的插值条件代入式(5-3)，可得到如下 P 个关于未知系数 ω_p 的线性方程组：

$$\begin{cases}\sum_{p=1}^{P}\omega_p\varphi(\|\boldsymbol{X}^1-\boldsymbol{X}^p\|)=d^1\\ \sum_{p=1}^{P}\omega_p\varphi(\|\boldsymbol{X}^2-\boldsymbol{X}^p\|)=d^2\\ \vdots\\ \sum_{p=1}^{P}\omega_p\varphi(\|\boldsymbol{X}^P-\boldsymbol{X}^p\|)=d^P\end{cases}\tag{5-4}$$

令 $\varphi_{ip}=\varphi(\|\boldsymbol{X}^i-\boldsymbol{X}^p\|)$，$i=1,2,\cdots,P$，$p=1,2,\cdots,P$，则上述方程组可改写为

$$\begin{pmatrix}\varphi_{11}&\varphi_{12}&\cdots&\varphi_{1P}\\ \varphi_{21}&\varphi_{22}&\cdots&\varphi_{2P}\\ \vdots&\vdots&\ddots&\vdots\\ \varphi_{P1}&\varphi_{P2}&\cdots&\varphi_{PP}\end{pmatrix}\begin{pmatrix}\omega_1\\ \omega_2\\ \vdots\\ \omega_P\end{pmatrix}=\begin{pmatrix}d_1\\ d_2\\ \vdots\\ d_P\end{pmatrix}\tag{5-5}$$

记式(5-5)为

$$\boldsymbol{\Phi W}=\boldsymbol{d}\tag{5-6}$$

式中，$\boldsymbol{\Phi}$ 表示元素为 φ_{ip} 的 $P\times P$ 矩阵；$\boldsymbol{W}$ 和 $\boldsymbol{d}$ 分别表示系数向量和期望输出向量。若 $\boldsymbol{\Phi}$ 为可逆矩阵，则可通过式(5-6)计算出系数向量 $\boldsymbol{W}$，即

$$\boldsymbol{W}=\boldsymbol{\Phi}^{-1}\boldsymbol{d}\tag{5-7}$$

Micchelli 定理给出了插值矩阵 $\boldsymbol{\Phi}$ 为可逆矩阵的条件：对于一大类函数，如果 $\boldsymbol{X}^1$，$\boldsymbol{X}^2,\cdots,\boldsymbol{X}^P$ 各不相同，则插值矩阵 $\boldsymbol{\Phi}$ 为可逆矩阵。满足上述条件的径向基函数通常有如下几种。

(1) 高斯(Gauss)函数。

$$\varphi(r)=\exp\left(-\frac{r^2}{2\sigma^2}\right)\tag{5-8}$$

(2) 反演 S 型(Reflected Sigmoidal)函数。

$$\varphi(r)=\frac{1}{1+\exp\left(\frac{r^2}{\sigma^2}\right)}\tag{5-9}$$

(3) 逆多二次(Inverse Multiquadrics)函数。

$$\varphi(r)=\frac{1}{(r^2+\sigma^2)^{1/2}}\tag{5-10}$$

式中，σ 表示基函数的扩展常数或宽度，基函数的宽度越小，则对应的选择性就越高。

5.1.3　正则化 RBF 神经网络结构

通过径向基函数解决插值问题的过程中，式(5-7)给出的计算方式存在不适定问题。由于径向基函数的数量与训练样本数量相等，当训练样本数远远大于物理过程中固有的自由度时，问题就称为不适定的，此时求解插值矩阵的逆矩阵时会不稳定。

正则化理论(Regularization Theory)是 Tikhonov 于 1963 年提出的一种解决不适定问题的方法,其基本思想是通过某些含有解的先验知识的非负的辅助泛函使解稳定。由于正则化理论的推理涉及大量泛函运算,因此不对其展开描述,在此直接给出正则化 RBF 神经网络的结构。

正则化 RBF 神经网络的结构如图 5.1 所示。其特征在于隐含层节点数等于输入样本数,隐含层节点的激活函数为 Green 函数,通常采用式(5-8)所示的 Gauss 函数,径向基函数的中心为所有的输入样本,各径向基函数采用统一的扩展常数。

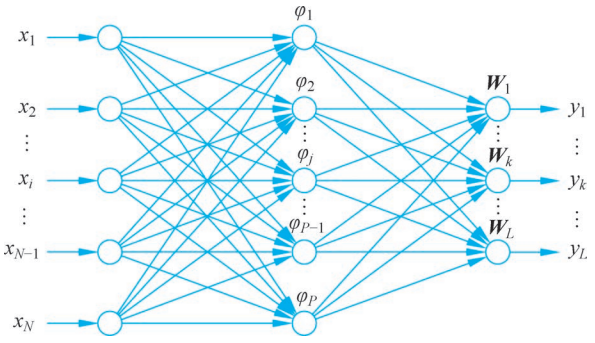

图 5.1 正则化 RBF 神经网络的结构

在图 5.1 所示的正则化 RBF 神经网络中,输入层节点数量为 N,隐含层节点数量为 P,输出层节点数量为 L。$\boldsymbol{X}=(x_1,x_2,\cdots,x_N)^{\mathrm{T}}$ 为网络的输入向量;$\varphi_j(\boldsymbol{X})(j=1,2,\cdots,P)$ 表示任意隐含层节点的激活函数,也称为基函数;$\boldsymbol{W}$ 为输出权值矩阵,其中 $\omega_{jk}(j=1,2,\cdots,P,k=1,2,\cdots,L)$,表示隐含层第 j 个节点与输出层第 k 个节点之间的权值;$\boldsymbol{Y}=(y_1,y_2,\cdots,y_L)^{\mathrm{T}}$ 为网络输出,输出层节点采用线性激活函数。

正则化 RBF 神经网络具有以下 3 个特点。①正则化网络是一种通用逼近器,只要有足够的隐节点,它就能以任意精度逼近紧集上的任意多元连续函数。②正则化网络具有最佳逼近特性,即任意给一个未知的非线性函数 f,总可以找到一组权值使得正则化网络对于 f 的逼近优于所有其他可能的选择。③正则化网络得到的解是最佳的,所谓“最佳”体现在同时满足对样本的逼近误差和逼近曲线的平滑性。

5.1.4 正则化 RBF 神经网络学习算法

正则化 RBF 神经网络的隐含层节点数即为样本数,基函数的数据中心即为样本自身,当采用正则化 RBF 神经网络进行训练时,只需要考虑扩展常数和输出节点的权值。

径向基函数的扩展常数可根据数据中心的分布而定,为了避免每一个径向基函数太尖或太平,可以将全部径向基函数的扩展常数设为

$$\sigma=\frac{d_{\max}}{\sqrt{2P}} \tag{5-11}$$

式中,$d_{\max}$ 为所选数据中心之间的最大距离;P 是样本数量。

网络输出节点的权值可以通过最小均方偏差算法(Least Mean Square,LMS)计算,LMS 的输入向量即为隐含层节点的输出向量$\boldsymbol{\Phi}$,权值调整公式为

$$\Delta\boldsymbol{W}_k=\eta(d_k-\boldsymbol{W}_k^{\mathrm{T}}\boldsymbol{\Phi})\boldsymbol{\Phi} \tag{5-12}$$

$\Delta \boldsymbol{W}_k$ 的各分量为

$$\Delta w_{jk} = \eta(d_k - \boldsymbol{W}_k^{\mathrm{T}}\boldsymbol{\Phi})\varphi_j \tag{5-13}$$

式中，$k=1, 2, \cdots, L$；$j=1, 2, \cdots, P$。

5.1.5　正则化 RBF 神经网络局限性

正则化 RBF 神经网络要求隐含层神经元个数为样本数，如果训练样本数量过大，则网络计算量也将大幅增加，从而导致计算效率过低。同时，当样本数量过大时，插值矩阵$\boldsymbol{\Phi}$ 是病态矩阵的概率也会提高，即$\boldsymbol{\Phi}$ 中的一个微小扰动将对计算结果 $\boldsymbol{W}$ 产生很大影响。

解决以上问题的方案是减少隐含层神经元的个数，同时优化网络参数的学习方法，以此实现对正则化 RBF 神经网络的改进，改进后的网络称之为广义 RBF 神经网络。

5.2　广义 RBF 神经网络


视频讲解

广义 RBF 神经网络与正则化 RBF 神经网络的区别主要在于隐含层节点的数量和网络参数的学习方法，本节将从模式可分性的角度阐述广义 RBF 神经网络的设计思想，并给出网络参数的学习方法。

5.2.1　模式可分性

根据感知器的基本特性可知，如果 N 维空间下的输入模式是线性可分的，那么总存在一个能用线性方程描述的超平面将输入模式进行划分；反之，如果输入模式不是线性可分的，则不存在相应的超平面将输入模式进行划分，此时需要将线性不可分问题转换为线性可分问题，这便是 Cover 定理。

Cover 定理可以定性地描述为：将复杂的模式分类问题非线性地投射到高维空间比投射到低维空间更可能是线性可分的。

设 $\boldsymbol{F}$ 为输入模式 $\boldsymbol{X}^p$（$p=1, 2, \cdots, P$）的集合，该集合内的所有模式可分为两类，分别属于集合 $\boldsymbol{F}_1$ 和 $\boldsymbol{F}_2$，集合 $\boldsymbol{F}$、$\boldsymbol{F}_1$ 和 $\boldsymbol{F}_2$ 满足关系：

$$\begin{cases} \boldsymbol{F}_1 \cup \boldsymbol{F}_2 = \boldsymbol{F} \\ \boldsymbol{F}_1 \cap \boldsymbol{F}_2 = \varnothing \end{cases} \tag{5-14}$$

若输入模式所在的空间内存在一个超曲面，使得分别属于 $\boldsymbol{F}_1$ 和 $\boldsymbol{F}_2$ 的模式能够分开，则称这些模式的二元划分关于该曲面是可分的。若该曲面为线性方程 $\boldsymbol{W}^{\mathrm{T}}\boldsymbol{X}=\boldsymbol{0}$ 确定的超平面，则称这些模式的二元划分关于该平面是线性可分的；反之，则称这些模式的二元划分是线性不可分的。

设有由一组函数构成的向量 $\varphi(\boldsymbol{X})=[\varphi_1(\boldsymbol{X}), \varphi_2(\boldsymbol{X}), \cdots, \varphi_M(\boldsymbol{X})]$，将原来 N 维空间的 P 个模式映射到新的 M 维空间（$M>N$），如果在 M 维空间存在 M 维向量 $\boldsymbol{W}$，使得

$$\begin{cases} \boldsymbol{W}^{\mathrm{T}}\varphi(\boldsymbol{X}) > 0, & \boldsymbol{X} \in \boldsymbol{F}_1 \\ \boldsymbol{W}^{\mathrm{T}}\varphi(\boldsymbol{X}) < 0, & \boldsymbol{X} \in \boldsymbol{F}_2 \end{cases} \tag{5-15}$$

则由线性方程 $\boldsymbol{W}^{\mathrm{T}}\varphi(\boldsymbol{X})=0$ 确定了 M 维空间中的一个分界超平面，这个超平面使得映射到

M 维空间中的 P 个点能够线性可分；而在 N 维空间下，则是存在一个超曲面使得原来在 N 维空间下非线性可分的 P 个模式分为两类，此时称原空间的 P 个模式是可分的。

在 RBF 神经网络中，将输入空间的模式非线性地映射到一个高维空间的方法是，设置一个隐含层，令 $\varphi(\boldsymbol{X})$ 为隐含层节点的激活函数，并使得隐含层节点数 M 大于输入节点数 N，使得在输入层 N 维空间下的线性不可分问题转换为隐含层 M 维空间下的线性可分问题。

5.2.2 广义 RBF 神经网络结构

广义 RBF 神经网络的基本思想是，用径向基函数作为隐含层神经元的“基”构成隐含层空间，隐含层空间的 M 个神经元对 P 个输入模式对应的向量进行变换，将低维的模式变换到高维的空间，使得低维空间的线性不可分问题转换为高维空间的线性可分问题。广义 RBF 神经网络结构如图 5.2 所示。

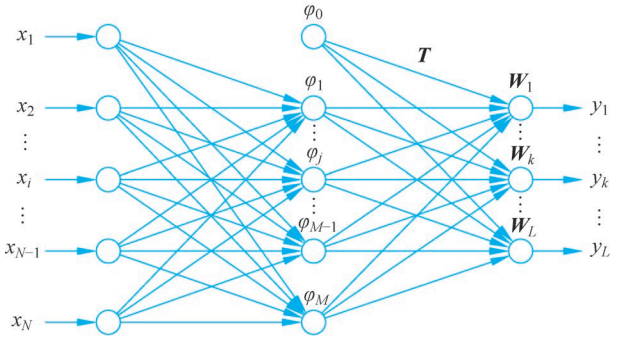

图 5.2 广义 RBF 神经网络结构

图 5.2 所示为 N-M-L 结构的广义 RBF 神经网络，网络具有 N 个输入节点，M 个隐含层节点，L 个输出节点，且 $N<M<P$。$\boldsymbol{X}=(x_1,x_2,\cdots,x_N)^{\mathrm{T}}$ 为网络的输入向量；$\varphi_j(\boldsymbol{X})$ $(j=1,2,\cdots,M)$ 表示任意隐含层节点的激活函数，也称为基函数，这里 φ_0 为 -1；$\boldsymbol{W}$ 为输出权值矩阵，$\boldsymbol{W}=(\boldsymbol{W}_1,\boldsymbol{W}_2,\cdots,\boldsymbol{W}_L)^{\mathrm{T}}$，其中，$\boldsymbol{W}_k=(\omega_{1k},\omega_{2k},\cdots,\omega_{Mk})$ 为隐含层 M 个节点到输出层第 k 个节点的权值组成的权向量，ω_{jk} $(j=1,2,\cdots,M,k=1,2,\cdots,L)$ 表示隐含层第 j 个节点与输出层第 k 个节点之间的权值；$\boldsymbol{T}=(T_1,T_2,\cdots,T_L)^{\mathrm{T}}$ 为输出层的阈值向量，其中，T_k $(k=1,2,\cdots,L)$ 表示输出层第 k 个节点的阈值；$\boldsymbol{Y}=(y_1,y_2,\cdots,y_L)^{\mathrm{T}}$ 为网络输出，输出层节点采用线性激活函数。

相比于正则化 RBF 神经网络，广义 RBF 神经网络的径向基函数中心不再限制在数据点上，各径向基函数的扩展常数也不再统一，均由训练算法决定；广义 RBF 神经网络的输出包含阈值参数，用于补偿基函数在样本集上的平均值与目标值平均值之间的差别。

在应用广义 RBF 神经网络解决实际问题时，有时隐含层节点的个数与网络输入模式对应的维数相同，换言之，就是将低维的模式变换到同维的空间内，也可将原来低维空间下的线性不可分问题转换为同维空间下的线性可分问题。下面以用广义 RBF 神经网络解决异或问题的例子来说明。

【例 5.1】　“异或”的真值表如表 5.1 所示，试用广义 RBF 神经网络实现“异或”功能。

表 5.1　“异或”的真值表

x_1	x_2	y
1	1	0
0	0	0
1	0	1
0	1	1

解：

1）问题分析

表 5.1 中的 4 个样本的输入分别为 $\boldsymbol{X}^1=\begin{pmatrix}1\\1\end{pmatrix}$、$\boldsymbol{X}^2=\begin{pmatrix}0\\0\end{pmatrix}$、$\boldsymbol{X}^3=\begin{pmatrix}1\\0\end{pmatrix}$、$\boldsymbol{X}^4=\begin{pmatrix}0\\1\end{pmatrix}$，输出分别为 $\boldsymbol{Y}^1=(0)$、$\boldsymbol{Y}^2=(0)$、$\boldsymbol{Y}^3=(1)$、$\boldsymbol{Y}^4=(1)$。“异或”问题是需要将上述 4 个样本分成两类，4 个样本在二维平面中的位置如图 5.3 所示。我们会发现，平面中的任何一条直线都不能将这两类样本分开，这是一个在原始二维空间下的线性不可分问题。

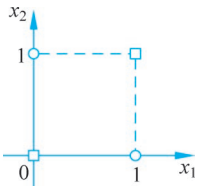

图 5.3　“异或”问题的线性不可分性示意图

2）建立 RBF 神经网络

由于样本有 2 个分量，因此输入节点设为 2。这里隐含层节点也设为 2，两个隐含层节点的径向基函数 $\varphi_1(x)$、$\varphi_2(x)$ 分别如式(5-16)、式(5-17)所示，两个基函数的数据中心分别是样本点 $\boldsymbol{X}^1$、$\boldsymbol{X}^2$。输出层节点设为 1 个，输出 0、1 代表两类样本。

$$\varphi_1(\boldsymbol{X})=\varphi(\boldsymbol{X}-\boldsymbol{X}^1)=\mathrm{e}^{-\|\boldsymbol{X}-\boldsymbol{X}^1\|^2} \tag{5-16}$$

$$\varphi_2(\boldsymbol{X})=\varphi(\boldsymbol{X}-\boldsymbol{X}^2)=\mathrm{e}^{-\|\boldsymbol{X}-\boldsymbol{X}^2\|^2} \tag{5-17}$$

构建的 RBF 神经网络如图 5.4 所示。4 个数据样本点输入 RBF 神经网络后，隐含层节点及输出层节点的输出如表 5.2 所示。将 4 个数据样本相应隐含层节点的输出标在隐含层空间如图 5.5 所示，我们可以看到 4 个数据样本在原来输入空间内线性不可分，通过变换到隐空间就变成了线性可分。

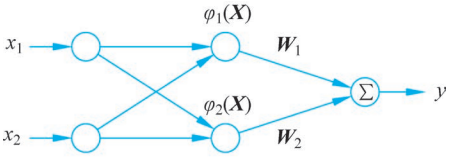

图 5.4　构建的 RBF 神经网络

表 5.2　“异或”问题 RBF 神经网络的隐含层节点及输出层节点的输出

x_1	x_2	$\varphi_1(\boldsymbol{X})$	$\varphi_2(\boldsymbol{X})$	y
1	1	1	0.1353	0
0	0	0.1353	1	0
1	0	0.3678	0.3678	1
0	1	0.3678	0.3678	1

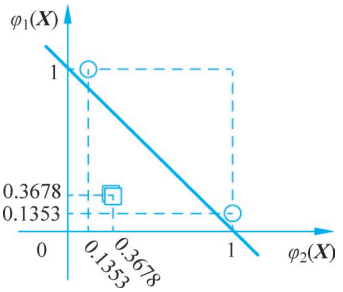

图 5.5 输入空间 4 个数据样本点映射到隐含层空间后的样本点分布图

像“异或”这类不太复杂的非线性模式分类问题，其 RBF 神经网络隐含层节点是 2 个，虽与样本的维数相等，但是问题也得到了解决。


视频讲解

5.2.3 广义 RBF 神经网络学习算法

广义 RBF 神经网络需要训练的参数包括隐含层节点数量、各径向基函数的数据中心和扩展常数，以及输出权值矩阵。这些参数的训练通常遵循由 Moody 和 Darken 于 1989 年提出的混合学习过程，该过程主要包括两个阶段：第一阶段为无监督的自组织学习阶段，其目的是为隐含层节点的径向基函数确定合适的数据中心和扩展常数；第二阶段是有监督学习阶段，其目的是训练网络输出层的权值。具体步骤如下。

1. 确定隐含层节点数量及隐含层节点径向基函数数据中心

广义 RBF 神经网络的隐含层节点数量和隐含层节点径向基函数数据中心的确定方法，可以依据 RBF 神经网络的训练样本采用聚类算法，如 SOM 聚类算法和 K-means 聚类算法等实现。

通过 4.2.2 节的 SOM 聚类算法得到的类别数、各类别对应的内星权向量分别作为广义 RBF 神经网络的隐含层节点数量 M，以及各隐含层节点径向基函数的数据中心 $\boldsymbol{c}_j$，$j=1,2,\cdots,M$。

通过 K-means 聚类算法得到的聚类类别数 M，以及各类别的几何中心向量分别作为广义 RBF 神经网络的隐含层节点数量 M，以及各隐含层节点径向基函数的数据中心。下面简要介绍 K-means 聚类算法，其过程如下。这里需要注意的是聚类类别 M 需要事先设定。

(1) 初始化。令 $k=0$，选择 M 个互不相同的向量 $\boldsymbol{c}_1(0),\boldsymbol{c}_2(0),\cdots,\boldsymbol{c}_M(0)$ 作为初始聚类中心。

(2) 计算各样本点与聚类中心点的距离：$\|\boldsymbol{X}^p-\boldsymbol{c}_j(k)\|$，其中 $p=1,2,\cdots P$；$j=1,2,\cdots,M$。

(3) 相似匹配。令 j^* 代表竞争获胜隐节点的下标，当

$$j^*(\boldsymbol{X}^p)=\min_j\|\boldsymbol{X}^p-\boldsymbol{c}_j(k)\|,\quad p=1,2,\cdots,P \tag{5-18}$$

时，$\boldsymbol{X}^p$ 被归为第 j^* 类，从而将全部样本划分为 M 个子集：$\boldsymbol{U}_1(k),\boldsymbol{U}_2(k),\cdots,\boldsymbol{U}_M(k)$，每个子集构成一个以聚类中心为典型代表的聚类域。

(4) 更新各类的聚类中心。令 $\boldsymbol{U}_j(k)$ 表示第 j 个聚类域，N_j 为第 j 个聚类域中的样本数，则

$$\boldsymbol{c}_j(k+1)=\frac{1}{N_j}\sum_{\boldsymbol{X}\in\boldsymbol{U}_j(k)}\boldsymbol{X} \tag{5-19}$$

(5) 令 $k=k+1$，转到第(2)步。重复上述过程，直到 $\boldsymbol{c}(k+1)=\boldsymbol{c}(k)$时停止训练。

2. 确定径向基函数的扩展常数

各聚类中心确定后，可根据各中心之间的距离确定对应径向基函数的扩展常数。令

$$d_j=\min_i\|\boldsymbol{c}_j-\boldsymbol{c}_i\|,\quad i\neq j \tag{5-20}$$

则扩展常数取

$$\sigma_j=\lambda d_j \tag{5-21}$$

式中，λ 为重叠系数。

3. 计算输出层的权值矩阵

输出层权值矩阵的计算方式可以采用 LMS 算法，参考正则化 RBF 神经网络的学习方法。除此之外，更为简洁的方法是采用伪逆直接计算。设输入为 $\boldsymbol{X}^p$ 时，第 j 个隐含层节点的输出为

$$\varphi_{pj}=\varphi(\|\boldsymbol{X}^p-\boldsymbol{c}_j\|) \tag{5-22}$$

式中，$p=1,2,\cdots,P$；$j=1,2,\cdots,M$，则隐含层节点的输出矩阵为

$$\hat{\boldsymbol{\Phi}}=[\varphi_{pj}]_{P\times M} \tag{5-23}$$

若 RBF 神经网络的待确定输出权值为 $\boldsymbol{W}=[\omega_{jk}]_{M\times L}$，则网络的输出为

$$F(\boldsymbol{X})=\hat{\boldsymbol{\Phi}}\boldsymbol{W} \tag{5-24}$$

令网络的输出等于数据集对应的输出标签 $\boldsymbol{d}$，则输出权值 $\boldsymbol{W}$ 可用$\hat{\boldsymbol{\Phi}}$ 的伪逆矩阵$\hat{\boldsymbol{\Phi}}^+$求出：

$$\boldsymbol{W}=\hat{\boldsymbol{\Phi}}^+\boldsymbol{d} \tag{5-25}$$

式中，$\hat{\boldsymbol{\Phi}}^+=(\hat{\boldsymbol{\Phi}}^{\mathrm{T}}\hat{\boldsymbol{\Phi}})^{-1}\hat{\boldsymbol{\Phi}}^{\mathrm{T}}$。

5.3　基于 MATLAB 的 RBF 神经网络应用案例

5.3.1　基于 MATLAB 的 RBF 神经网络案例——数据拟合


视频讲解

1. 实验描述及数据准备

本实验利用 RBF 神经网络对 MATLAB 自定义数据集 simplefit_dataset 实现离散数据的拟合。

自定义包含 36 个输入-输出数据对的 simplefit_dataset 数据集，绘制原始数据样本图如图 5.6 所示。数据集中包含两个变量，分别是输入向量 dataset_input 和输出向量 dataset_output，这两个向量的数据维度均为 1×36，即向量中包含 36 个一维数据，具体数据见下述代码。

```
% 定义原始数据
dataset_input = -9:26;
dataset_output = [129, -32, -118, -138, -125, -97, -55, -23, -4,2,1, -31, -72, -121,
-142, -174, -155, -77,140, -43, -149, -169, -186, -98, -78, -34, -8,15,4, -67, -93,
-152, -173, -195, -156, -45];
```

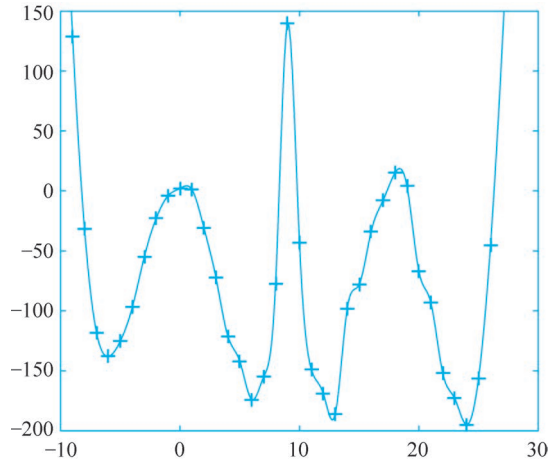

图 5.6　原始数据样本图

2. RBF 神经网络设计

本实验可以通过调用 MATLAB 工具箱函数 newrb 构建 RBF 神经网络来完成数据拟合。为了使读者更好地理解径向基函数的作用，图 5.7 展示了径向基函数 a1 的基本特性，并给出 3 个径向基函数 a1、a2、a3 及其加权求和的效果如图 5.8 所示，方便读者理解 RBF 神经网络的工作过程，径向基函数通过函数 radbas 实现。

3 个隐含层节点的径向基函数 a1、a2、a3 都选择高斯函数，径向基函数 a1 的数据中心为 0，对数据的处理程序如下。

```
x = -3:.1:3;
a1 = radbas(x);
figure(2)
plot(x,a1)
title('Radial Basis Transfer Function');
xlabel('Input x');
ylabel('Output a1');
```

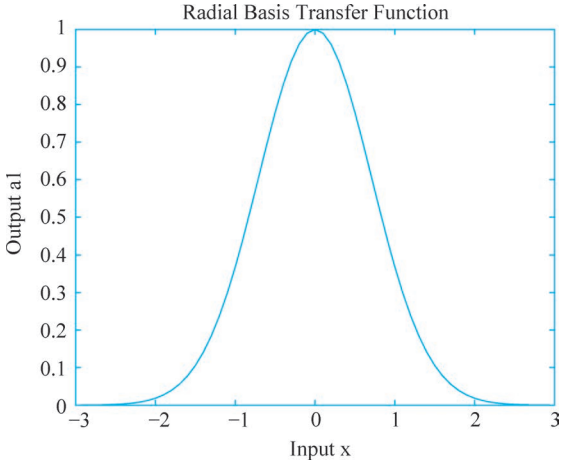

图 5.7　径向基函数 a1

通过观察 3 个径向基函数 a1、a2、a3 加权求和成 a4 的效果可以看出，如果选择合适的径向基函数宽度以及加权系数等，则可以形成新的曲线以拟合样本数据。

```
a2 = radbas(x - 1.5);
a3 = radbas(x + 2);
a4 = a1 + a2 * 1 + a3 * 0.5;
plot(x,a1,'k:',x,a2,'k -- ',x,a3,'k - ',x,a4,'b - ','LineWidth',1.5)
title('Weighted Sum of Radial Basis Transfer Functions');
xlabel('Input x');
ylabel('Output a4');
legend('a1','a2','a3','a4')
```

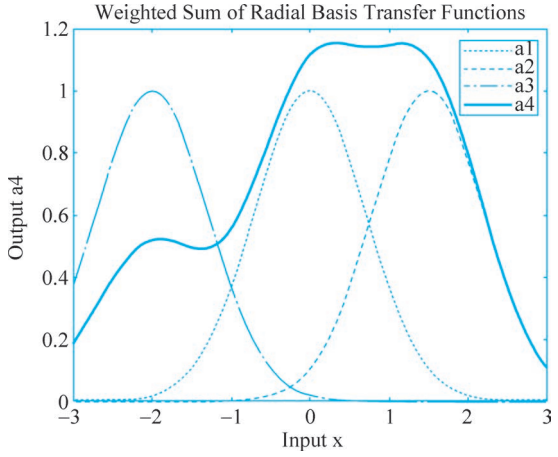

图 5.8　3 个径向基函数 a1、a2、a3 及其加权求和的效果

除了采用径向基函数实现数据拟合，也可以基于输入/输出数据，直接利用 newrb 函数进行设计，例如产生一个新的 RBF 神经网络：net = newrb(dataset_input,dataset_output,eg,sc)，其中 dataset_input、dataset_output 分别代表输入和输出；eg 为训练目标误差；sc 为径向基函数的扩展常数，sc 越小，径向基函数宽度越窄，每个径向基函数覆盖的范围越小，拟合所需要的径向基函数越多。下面使用本案例中的输入/输出数据对比不同 sc 的影响，添加如下程序。

```
eg = 0.02;
sc1 = 10; net = newrb(dataset_input,dataset_output,eg,sc1);
Y1 = net(dataset_input);
sc2 = 100; net = newrb(dataset_input,dataset_output,eg,sc2);
Y2 = net(dataset_input);
sc3 = 1; net = newrb(dataset_input,dataset_output,eg,sc3);
Y3 = net(dataset_input);
plot(dataset_input,Y1,'r - ',dataset_input,Y2,'k -- ',dataset_input,Y3,'b - .');
legend('sample','sc = 10','sc = 100','sc = 1')
hold off;
```

不同扩展常数下 RBF 的拟合效果如图 5.9 所示。

本实验调用 MATLAB 工具箱函数 newrb 构建 RBF 神经网络来完成数据拟合。newrb 工具箱函数可根据网络训练情况自动增加隐含层神经元个数，因此无须设置网络的隐含层节点个数，仅需确定 RBF 神经网络的输入层与输出层节点个数、训练目标误差、扩展常数、隐含层最大神经元个数以及每次迭代增加的隐含层神经元个数。

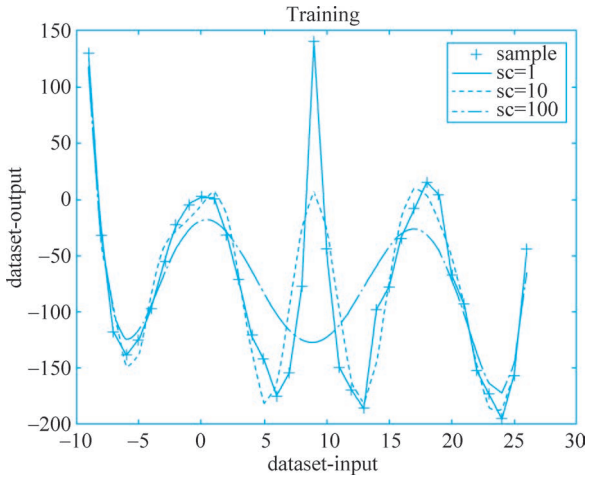

图 5.9 不同扩展常数下 RBF 的拟合效果

本实验中，RBF 神经网络输入层与输出层节点个数均为 1 个，扩展常数设置为 2，训练目标误差设置为 10^{-6}。隐含层的节点个数由 newrb 函数自动确定，根据用户设置的目标误差，自动调整隐含层节点个数，直至网络训练误差达到用户设置的目标误差，或者隐含层节点数达到最大样本数为止。

3. RBF 神经网络实现

利用 MATLAB 软件实现 RBF 神经网络对 simplefit_dataset 数据集进行拟合的步骤如下。

1）数据集导入

```
load simplefit_dataset;                    % 导入数据集
```

2）参数设置

```
spread = 2;                    % 设置网络扩展常数为 2
goal_mse = 10^-6;              % 设置网络训练目标误差为 10^-6
max_neurons = size(dataset_output,2); % 设置隐含层节点数最大值为样本个数，这里 size 函数中
                               % 参数值 2 表示的含义是，该函数返回输出向量 dataset_output 的列数
```

3）RBF 神经网络训练

```
% 利用输入向量 dataset_input 和输出向量 dataset_output 训练 RBF 神经网络
net = newrb(dataset_input, dataset_output, goal_mse, spread,max_neurons,1);
% 这里 newrb 函数中参数值 1 表示的含义是，RBF 神经网络训练中每次迭代时增加 1 个隐含层神经元
```

4）获取网络对输入向量的拟合输出

```
Output_sim = sim(net,dataset_input);                    % 网络输出
```

4. 仿真结果及分析

在 RBF 神经网络训练时，可通过 newrb 窗口观察网络训练过程如图 5.10 所示。从图 5.10 中可知在第 35 次迭代训练中，网络的训练精度达到预设值，网络隐含层共包含 35 个节点。

绘制 RBF 神经网络拟合数据与原始数据对比图，以便观察网络拟合效果。如图 5.11 所示，原始数据与拟合数据保持一致，说明使用 RBF 神经网络得到了理想的数据拟合效果。

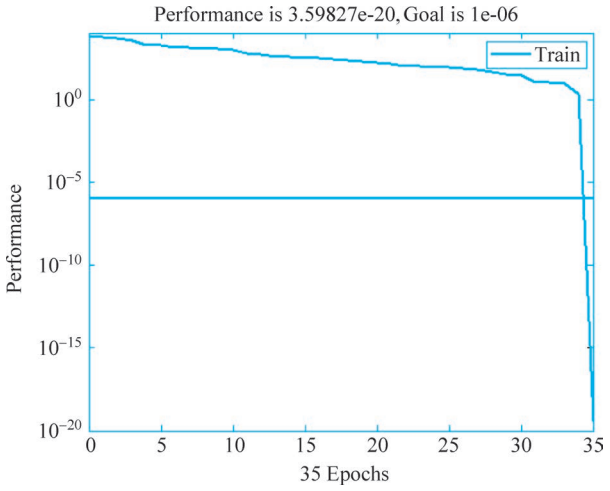

图 5.10　网络训练过程

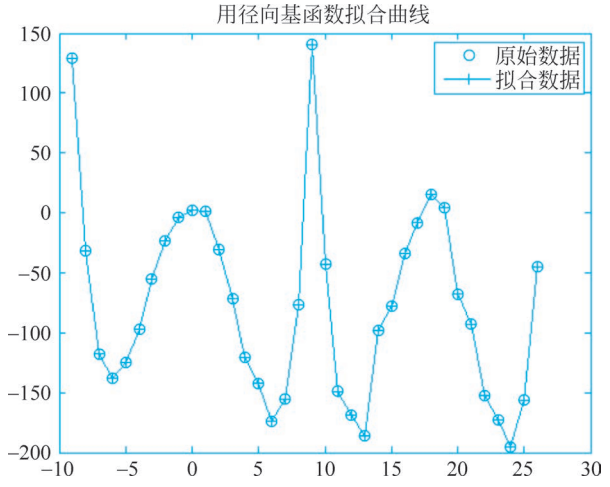

图 5.11　网络拟合效果

5.3.2　基于 MATLAB 的 RBF 神经网络案例——小麦种子分类

视频讲解

1. 案例分析

不同品种的植物种子具有不同的特征，如面积、周长、籽粒长度、籽粒宽度等。根据不同的籽粒特征可以筛选出不同品种的植物种子，对于种植效率的提高具有非常重要的实际意义。本案例采用基于 RBF 神经网络的分类方法对三类种子进行分类实验，通过 RBF 神经网络确定种子特征与种子类别之间的非线性映射关系。

本案例中小麦种子数据集 seeds_dataset 来源于加州大学欧文分校（University of California，Irvine，UCI）机器学习数据库，该数据集中包括三种不同小麦品种的籽粒：卡马、罗莎和加拿大小麦。设计基于 RBF 神经网络的分类模型对小麦种子进行分类，从 210 组数据样本中随机选取 150 组数据构成训练集完成 RBF 神经网络分类模型的训练，剩余 60 组数据组成测试集，进一步测试分类模型的泛化能力。

2. 数据集分析

本案例选用的 seeds_dataset.txt 数据集包含 3 种不同的小麦，每种小麦有 70 组样本，因此共包含 210 组数据样本。每组数据样本包含 7 个小麦几何特征参数及 1 个对应小麦种类标签编号，因此数据集为一个 8×210 的矩阵形式，部分样本数据如表 5.3 所示。

表 5.3 seeds_dataset 部分样本数据

样本编号	面积	周长	紧密度	籽粒长度	籽粒宽度	不对称系数	籽粒槽的长度	标签
1	15.26	14.84	0.871	5.763	3.312	2.221	5.22	1
2	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
3	14.29	14.09	0.905	5.291	3.337	2.699	4.825	1
4	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
5	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
71	17.63	15.98	0.8673	6.191	3.561	4.076	6.06	2
72	16.84	15.67	0.8623	5.998	3.484	4.675	5.877	2
73	17.26	15.73	0.8763	5.978	3.594	4.539	5.791	2
74	19.11	16.26	0.9081	6.154	3.93	2.936	6.079	2
75	16.82	15.51	0.8786	6.017	3.486	4.004	5.841	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
141	13.07	13.92	0.848	5.472	2.994	5.304	5.395	3
142	13.32	13.94	0.8613	5.541	3.073	7.035	5.44	3
143	13.34	13.95	0.862	5.389	3.074	5.995	5.307	3
144	12.22	13.32	0.8652	5.224	2.967	5.469	5.221	3
145	11.82	13.4	0.8274	5.314	2.777	4.471	5.178	3

本案例数据集中每组小麦样本含有 7 个小麦籽粒的几何参数特征：面积、周长、紧密度、籽粒长度、籽粒宽度、不对称系数以及籽粒槽的长度。共有 3 种不同的小麦：卡马、罗莎和加拿大小麦，对应标签编号分别为 1、2、3。

3. 小麦种子分类 RBF 神经网络设计

RBF 神经网络是标准三层网络结构，包含输入层、隐含层和输出层。基于 RBF 神经网络的小麦种子分类模型的网络设计包含结构设计和参数设计，本案例通过 MATLAB 自带的神经网络工具箱提供的 newrb 函数创建 RBF 神经网络。

1）网络结构设计

输入层、输出层节点数取决于数据集每组样本的输入、输出特征参数个数，即输入向量的维数和输出向量的维数。在本案例中输入节点个数为 7，输出节点个数为 1。隐含层的节点个数由 newrb 函数自动确定，根据用户设置的训练目标误差，自动调整隐含层节点个数，直至网络训练误差达到用户设置的训练目标误差，或者隐含层节点数达到最大样本数为止。

2）参数设计

调用 MATLAB 神经网络工具箱的 newrb 函数创建 RBF 神经网络，其径向基函数为 MATLAB 默认函数，即高斯函数，径向基函数的扩展常数设置为默认值 1。

4. 小麦种子分类 RBF 神经网络实现

本案例探讨的基于 RBF 神经网络的小麦种子分类模型将通过 MATLAB 神经网络工具箱函数来仿真实现。

1）导入数据集

导入本案例数据集，该数据集中包含了 210 组数据样本。随机生成一组 1～210 的序列，将数据集样本数据按照随机序列重新排序划分训练集与测试集，前 150 组样本数据作为 RBF 神经网络训练集数据，剩余 60 组样本数据作为 RBF 神经网络测试集数据，分别生成样本数据特征输入矩阵和样本数据标签输出矩阵。

```
load ('seeds_dataset.txt');                    % 加载样本数据
data = seeds_dataset';                         % 重命名数据集
rank = randperm(210)                           % 生成一个 1～210 的随机序列
data1 = data(:,rank);                          % 按照 rank 顺序重新排列，生成数据集
train_x = data1(1:7,1:150);                    % 生成训练集样本数据特征输入矩阵
train_t = data1(8,1:150);                      % 生成训练集样本数据标签输出矩阵
test_x = data1(1:7,151:210);                   % 生成测试集样本数据特征输入矩阵
test_t = data1(8,151:210);                     % 生成测试集样本数据标签输出矩阵
```

2）搭建 RBF 神经网络并进行网络训练

初始化 RBF 神经网络参数，通过 MATLAB 神经网络工具箱提供的 newrb 函数可创建 RBF 神经网络，以训练集样本数据训练 RBF 神经网络小麦种子分类模型，设置 RBF 神经网络训练的目标误差为 1e-8，设置径向基函数扩展常数为 1，隐含层节点个数最大设置为训练集样本个数 150，在训练过程中以“1”为单位逐步递增隐含层神经元个数直到训练误差低于目标误差。

```
net = newrb(train_x,train_t,1e-8,1,150,1);                    % 创建 RBF 神经网络
```

3）测试集数据样本验证 RBF 神经网络模型分类效果

根据 RBF 神经网络小麦种子分类模型的输出结果，基于测试集中每类小麦的数量计算测试集分类的准确率及每类小麦各自分类的正确率。

```
y2 = net(test_x);                    % 测试集样本数据 RBF 神经网络输出
perf2 = perform(net,test_t,y2);
for j = 1:length(y2)
   if y2(j)<=1
      y2(j) = 1;
   elseif y2(j)<=2
       y2(j) = 2;
   else
       y2(j) = 3;
   end
end
acc2 = sum(y2 == test_t)/60;

%% 结果显示
total_A = 70;
total_B = 70;
total_C = 70;
count_A = length(find(train_t == 1));
count_B = length(find(train_t == 2));
count_C = length(find(train_t == 3));
number_A = length(find(test_t == 1));
```

```
number_B = length(find(test_t == 2));
number_C = length(find(test_t == 3));
number_A_sim = length(find(y2 == 1 &test_t == 1));
number_B_sim = length(find(y2 == 2 &test_t == 2));
number_C_sim = length(find(y2 == 3 &test_t == 3));
disp(['种子总数:' num2str(210) '第一类:' num2str(total_A) '第二类:' num2str(total_B) '第三
类:' num2str(total_C)]);
disp(['训练集种子总数:' num2str(150) '第一类:' num2str(count_A) '第二类:' num2str(count_B)
'第三类:' num2str(count_C)]);
disp(['测试集种子总数:' num2str(60) '第一类:' num2str(number_A) '第二类:' num2str(number_B)
'第三类:' num2str(number_C)]);
fprintf('测试集分类准确率:%f\n',acc2);
disp(['第一类分类正确:' num2str(number_A_sim) '错误:' num2str(number_A - number_A_sim) '正
确率 p1 = ' num2str(number_A_sim/number_A * 100) '%']);
disp(['第二类分类正确:' num2str(number_B_sim) '错误:' num2str(number_B - number_B_sim) '正
确率 p2 = ' num2str(number_B_sim/number_B * 100) '%']);
disp(['第三类分类正确:' num2str(number_C_sim) '错误:' num2str(number_C - number_C_sim) '正
确率 p3 = ' num2str(number_C_sim/number_C * 100) '%']);
```

5. 实验结果分析

经过 RBF 神经网络小麦种子分类模型训练后，测试集样本数据的分类结果如图 5.12 所示。

```
种子总数：210  第一类：70  第二类：70  第三类：70
训练集种子总数：150  第一类：47  第二类：51  第三类：52
测试集种子总数：60  第一类：23  第二类：19  第三类：18
测试集分类准确率：0.800000
第一类分类正确：16  错误：7  正确率p1=69.5652%
第二类分类正确：16  错误：3  正确率p2=84.2105%
第三类分类正确：16  错误：2  正确率p3=88.8889%
```

图 5.12 测试集样本数据的分类结果

通过测试集样本数据对 RBF 神经网络小麦种子分类模型效果进行验证，可以看出 3 种小麦种子能被 RBF 神经网络分类模型较为准确地分类，表明基于 RBF 神经网络的分类模型具有良好的泛化能力及分类效果。

5.3.3 基于 MATLAB 的 RBF 神经网络案例——人口数量预测

1. 应用案例分析

我国是一个人口大国，人口问题始终是制约发展的关键因素之一。科学合理的人口数量预测，有助于国家最大效率地分配社会资源，制定更加合理的发展建设规划，实现整个社会的良性循环。

本案例以中国 1949—2013 年的人口统计数据为基础，构建以 n 年～$n+4$ 年（$n \in [1949,2008]$）人口数量为输入、以第 $n+5$ 年人口数量为输出的样本集，并划分训练样本与测试样本。建立基于 RBF 神经网络的人口数量预测模型，使训练后的模型可根据历年人口数量对未来人口数量做出预测，最后利用测试集测试模型泛化能力。

2. 数据集准备

本案例所用数据集来自中国人口统计年鉴，其中包含了 1949—2013 年中国年人口统计数量（单位：万），详细数据如表 5.4 所示。

表 5.4　1949 年—2013 年中国年人口统计数量　（单位：万）

年份	人口数量	年份	人口数量	年份	人口数量	年份	人口数量	年份	人口数量
1949	54 167	1962	67 295	1975	92 420	1988	111 026	2001	127 627
1950	55 196	1963	69 172	1976	93 717	1989	112 704	2002	128 453
1951	56 300	1964	70 499	1977	94 974	1990	114 333	2003	129 227
1952	57 482	1965	72 538	1978	96 259	1991	115 823	2004	129 988
1953	58 796	1966	74 542	1979	97 542	1992	117 171	2005	130 756
1954	60 266	1967	76 368	1980	98 705	1993	118 517	2006	131 448
1955	61 465	1968	78 534	1981	100 072	1994	119 850	2007	132 129
1956	62 828	1969	80 671	1982	101 654	1995	121 121	2008	132 802
1957	64 653	1970	82 992	1983	103 008	1996	122 389	2009	134 480
1958	65 994	1971	85 229	1984	104 357	1997	123 626	2010	135 030
1959	67 207	1972	87 177	1985	105 851	1998	124 761	2011	135 770
1960	66 207	1973	89 211	1986	107 507	1999	125 786	2012	136 460
1961	65 859	1974	90 859	1987	109 300	2000	126 743	2013	137 510

本案例以 n 年～$n+4$ 年($n\in[1949,2008]$)人口数量为输入，以第 $n+5$ 年人口数量为输出，最终得到 60 组输入样本及期望输出样本。

3. 人口数量预测 RBF 神经网络结构设计

本实验调用 MATLAB 工具箱函数 newrb 构建 RBF 神经网络来完成人口数量预测，newrb 工具箱函数可根据网络训练情况自动增加隐含层神经元个数，因此无须设置 RBF 神经网络的隐含层节点个数，仅需确定 RBF 神经网络的输入层与输出层节点个数、训练目标误差、扩展常数。

本实验中，以 n 年～$n+4$ 年($n\in[1949,2008]$)人口数量为网络输入，以第 $n+5$ 年为网络期望输出。因此，设计 RBF 神经网络输入层节点为 5 个，输出层节点为 1 个，扩展常数设置为 2.3，训练目标误差设置为 10^{-6}。

4. 人口数量 RBF 神经网络预测模型的实现

下面介绍 RBF 神经网络的人口数量预测模型的实现过程，其详细步骤如下。

1）数据集导入及数据划分

(1) 首先导入人口数据，形成由 60 个人口数据组成的行向量。

```
data = xlsread('Population_data.xlsx');                    % 导入人口数据
```

(2) 然后生成输入/输出对应的样本集。

```
lag = 5;                          % 定义阶数，用前 lag 个人口数据作为样本的输入
iinput = data;                    % iinput 为原始序列
n = length(iinput);               % 获取 iinput 的长度
inputs = zeros(lag,n - lag);      % 定义维度为 5 × 60 的输入样本矩阵
for i = 1:n - lag                 % 用 i～i + 4 的数据作为样本输入，循环构建样本集的输入矩阵
    inputs(:,i) = iinput(i:i + lag - 1)';
end
targets = data(lag + 1:end);      % 从 data 数组第 6 个数据起到最后一个数据是样本集输出向量
```

(3) 最后按照每隔 6 个样本抽取一个样本为测试样本的方式，即以 6∶1 的比例划分训

练集和测试集，分别得到 52 个训练样本及 8 个测试样本。

```
k = 1;j = 1;z = 1;
for i = 1:size(inputs,2)
  if k < 7
  input_train(:,z) = inputs(:,i);                    % 训练样本的输入矩阵
  output_train(z) = targets(i);                      % 训练样本的输出矩阵
  k = k + 1;
  z = z + 1;
  else if k == 7
    input_test(:,j) = inputs(:,i);                   % 测试样本的输入矩阵
    output_test(j) = targets(i);                     % 测试样本的输出矩阵
    k = 1;
    j = j + 1;
  end
end
```

2）数据归一化处理

首先对训练集与测试集分别按照式(5-26)和式(5-27)进行归一化处理：

$$x^r_{\text{mid}} = \frac{x^r_{\max} + x^r_{\min}}{2}, \quad r = 1,2,\cdots,5 \tag{5-26}$$

$$\hat{x}_{nr} = \frac{x_{nr} - x^r_{\text{mid}}}{(x^r_{\max} - x^r_{\min})/2}, \quad n = 1,2,\cdots,m \tag{5-27}$$

式(5-26)中，x^r_{mid} 表示第 r 个分量数据变化范围内的中间值；$x^r_{\max}$ 和 $x^r_{\min}$ 分别表示第 r 个分量数据的最大值和最小值。式(5-27)中，x_{nr} 表示原始样本集中第 n 个样本的第 r 个分量；$\hat{x}_{nr}$ 表示 x_{nr} 归一化的结果。

```
% % 训练集输入数据归一化,归一化到[ - 1,1]
xmid = (137510 + 54167)/2;
for i = 1:size(input_train,1)
  for j = 1:size(input_train,2)
    inputn(i,j) = (input_train(i,j) - xmid)/((137510 - 54167)/2); % x = (x - xmid)/((xmax -
                                                                  % xmin)/2)
  end
end
% % 训练集输出数据归一化
for i = 1:size(input_train,2)
       outputn(i) = (output_train(i) - xmid)/((137510 - 54167)/2);
end
% % 测试集输入数据归一化
for i = 1:size(input_test,1)
  for j = 1:size(input_test,2)
      inputn_test(i,j) = (input_test(i,j) - xmid)/((137510 - 54167)/2);
  end
end
```

3）参数设置

设置网络的训练精度、学习率、最大迭代次数等参数并初始化网络各层结构。

```
max_neurons = size(inputn,2);                        % 设置网络隐含层节点数最大值为训练样本个数
spread = 2.3;                                        % 设置网络扩展常数为 2.3
goal_mse = 0.00001;                                  % 设置网络训练目标误差为 10^-5
```

4）网络训练

```
% 利用训练样本输入矩阵 inputn 和输出矩阵 outputn 训练 RBF 神经网络
net = newrb(inputn, outputn, goal_mse, spread,max_neurons,10);
```

5）网络测试

```
% 将测试集输入网络,得到各测试样本的网络输出向量
testNetworkOutput1 = sim(net,inputn_test);
```

5. 仿真结果及分析

利用 MATLAB 工具箱函数 newrb 训练人口数量预测模型时，可通过 newrb 窗口观察网络训练过程，网络训练误差变化如图 5.13 所示，可知在第 20 次迭代训练中，网络的训练精度达到预设值，网络隐含层共包含 20 个节点。

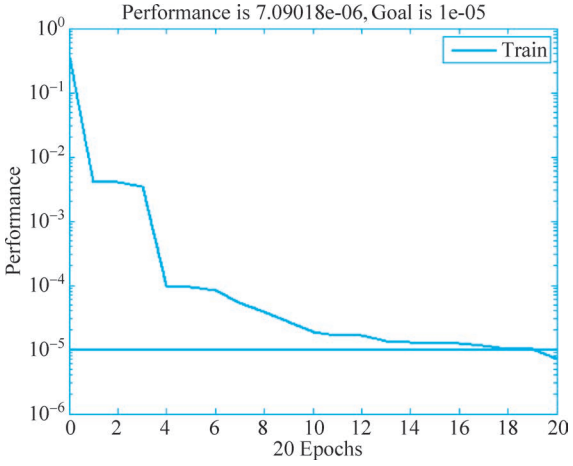

图 5.13　网络训练误差变化

训练集样本训练效果如图 5.14 所示，由图可知网络对训练样本的预测结果与其真实值吻合度高。网络测试结果如图 5.15 所示，由图可知大部分样本的预测值与真实值较为接近，网络测试的平均相对误差为 0.70%，泛化能力较强。

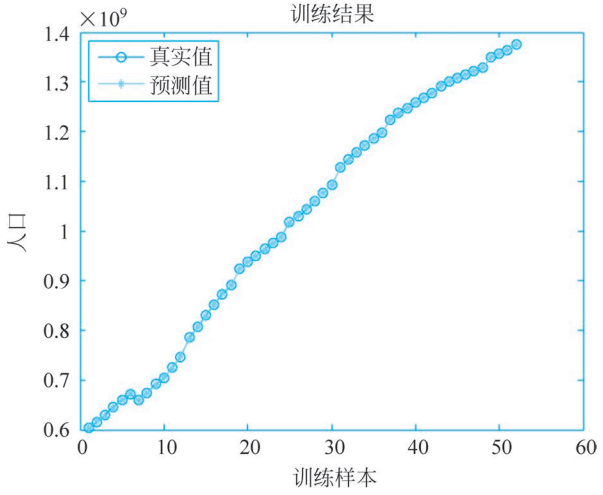

图 5.14　训练集样本训练效果

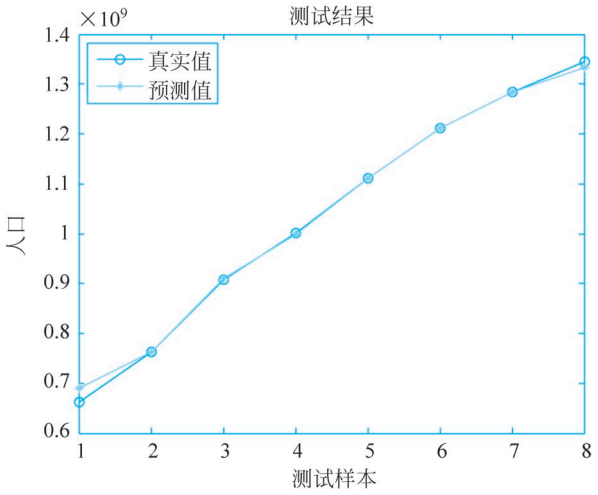

图 5.15 网络测试结果

5.3.4 基于 MATLAB 的 RBF 神经网络案例——地下水位预测

1. 案例分析

地下水系统是一个复杂的非线性、随机系统，影响地下水位变化的因素包括降水、气温、蒸发等，地下水的变化有一定的年度周期性，但又逐年波动。地下水位的变化可以在一定程度上反映地下水系统内部的变化情况，通过对地下水位的预测，能够有效地获取地下漏斗区的发展信息，更合理地利用和管理地下水资源，为实现水位调控打下基础。

由于 RBF 神经网络训练速度快，具有很强的非线性映射能力，因此本案例利用 RBF 神经网络建立影响地下水位的多个因素与地下水位之间的数学关系模型，构建基于 RBF 神经网络的地下水位预测模型。对构建出的地下水位预测模型进行训练，训练后使用测试集样本数据测试模型的预测精度及泛化能力，输出结果与期望值拟合程度越高则说明该模型的性能越好。最终目标是实现对地下水位的预测，得出较为准确可靠的预测数据。

2. 数据集分析

本案例数据集采用滦河某观测站 24 个月的地下水位检测样本数据，其中影响地下水位变化的因素分别有河道流量、气温、饱和差、降水量和蒸发量 5 个因素。因此本案例数据集共包含 24 组样本数据，每组样本数据中包含 5 个影响因素的数据以及 1 个相应的地下水位值，5 个影响因素数据作为一组样本数据的 RBF 神经网络特征输入量，相应的地下水位值则为 RBF 神经网络的期望输出量，样本数据集如表 5.5 所示。

表 5.5 样本数据集

序号	河道流量/$m^3 \cdot s^{-1}$	气温/℃	饱和差/hPa	降水量/mm	蒸发量/mm	水位/m
1	1.5	−10	1.2	1	1.2	6.92
2	1.8	−10	2	1	0.8	6.97
3	4	−2	2.5	6	2.4	6.84
4	13	10	5	30	4.4	6.5
5	5	17	9	18	6.3	5.75
6	9	22	10	13	6.6	5.54

续表

序号	河道流量/$m^3 \cdot s^{-1}$	气温/℃	饱和差/hPa	降水量/mm	蒸发量/mm	水位/m
7	10	23	8	29	5.6	6.63
8	9	21	6	74	4.6	5.62
9	7	15	5	21	2.3	5.96
10	9.5	8.5	5	15	3.5	6.3
11	5.5	0	6.2	14	2.4	6.8
12	12	−8.5	4.5	11	0.8	6.9
13	1.5	−11	2	1	1.3	6.7
14	3	−7	2.5	2	1.3	6.77
15	7	0	3	4	4.1	6.67
16	19	10	7	0	3.2	6.33
17	4.5	18	10	19	6.5	5.82
18	8	21.5	11	81	7.7	5.58
19	57	22	5.5	186	5.5	5.48
20	35	19	5	114	4.6	5.38
21	39	13	5	60	3.6	5.51
22	23	6	3	35	2.6	5.84
23	11	1	2	4	1.7	6.32
24	4.5	−2	1	6	1	6.56

3. RBF 神经网络地下水位预测模型设计

RBF 神经网络是标准三层结构，包含输入层、隐含层和输出层。RBF 神经网络的地下水位预测模型的网络设计包含结构设计和参数设计，本案例的 RBF 神经网络模型通过 MATLAB 神经网络工具箱提供的 newrb 函数进行创建。

1）结构设计

RBF 神经网络的输入层和输出层节点数与数据集中每个样本的输入和输出向量维度相对应。在本案例中输入节点个数为 5 个，输出节点个数为 1 个。通过 MATLAB 神经网络工具箱提供的 newrb 函数可创建 RBF 神经网络，隐含层的节点个数是不确定的，newrb 函数根据用户设置的目标误差，向网络中不断添加隐含层节点，直到训练误差低于目标误差为止，并且隐含层节点最大个数为样本个数。

2）参数设计

通过 MATLAB 神经网络工具箱提供的 newrb 函数创建 RBF 神经网络，径向基函数为 MATLAB 默认函数，RBF 神经网络的径向基函数扩展常数设置为 1。

4. RBF 神经网络地下水位预测模型实现

下面介绍基于 RBF 神经网络的地下水位预测模型的实现过程，通过调用 MATLAB 工具箱函数的 newrb 来构建网络模型，详细步骤如下。

1）导入数据集及数据预处理

（1）导入本案例数据集。数据集中包含 24 组数据样本，每组数据样本包含 5 个影响地下水位的因素数据，导入后得到大小为 5×24 的输入矩阵 $\boldsymbol{x}$；将每组数据样本对应的地下水位值作为网络的期望输出数据，得到大小为 1×24 的期望输出矩阵 $\boldsymbol{y}$。将 $\boldsymbol{x}$、$\boldsymbol{y}$ 矩阵第 6～24 列共 19 组样本数据作为 RBF 神经网络地下水位预测模型的训练集，将 $\boldsymbol{x}$、$\boldsymbol{y}$ 矩阵第

1～5 列共 5 组样本数据作为 RBF 神经网络地下水位预测模型的测试集。

```
%将数据集中影响地下水位的5个因素数据作为神经网络特征输入量存储在x矩阵中
x = [ 1.5, 1.8, 4.0, 13.0, 5.0, 9.0, 10.0, 9.0, 7.0, 9.5, 5.5, 12.0,...
   1.5, 3.0, 7.0, 19.0, 4.5, 8.0, 57.0, 35.0, 39.0, 23.0, 11.0, 4.5;
   -10.0, -10.0, -2.0, 10.0, 17.0, 22.0, 23.0, 21.0, 15.0, 8.5, 0, -8.5,...
   -11.0, -7.0, 0, 10.0, 18.0, 21.5, 22.0, 19.0, 13.0, 6.0, 1.0, -2.0;
   1.2, 2.0, 2.5, 5.0, 9.0, 10.0, 8.0, 6.0, 5.0, 5.0, 6.2, 4.5,...
   2.0, 2.5, 3.0, 7.0, 10.0, 11.0, 5.5, 5.0, 5.0, 3.0, 2.0, 1.0;
   1.0, 1.0, 6.0, 30.0, 18.0, 13.0, 29.0, 74.0, 21.0, 15.0, 14.0, 11.0,...
   1.0, 2.0, 4.0, 0, 19.0, 81.0, 186.0, 114.0, 60.0, 35.0, 4.0, 6.0;
   1.2, 0.8, 2.4, 4.4, 6.3, 6.6, 5.6, 4.6, 2.3, 3.5, 2.4, 0.8,...
   1.3, 1.3, 4.1, 3.2, 6.5, 7.7, 5.5, 4.6, 3.6, 2.6, 1.7, 1.0
   ];
%将数据集中相应的地下水位值作为神经网络期望输出量存储在y矩阵中
y = [6.92, 6.97, 6.84, 6.5, 5.75, 5.54, 5.63, 5.62, 5.96, 6.3, 6.8, 6.9,...
   6.7, 6.77, 6.67, 6.33, 5.82, 5.58, 5.48, 5.38, 5.51, 5.84, 6.32, 6.56];
trainx0 = x(1:5, 6:24);                %划分训练集特征输入量
trainy0 = y(6:24);                     %划分训练集期望输出量
testx = x(1:5,1:5);                    %划分测试集特征输入量
   testy = y(1:5);                     %划分测试集期望输出量
```

(2) 数据预处理。由于训练集样本数据量较小,为了提高 RBF 神经网络地下水位预测模型预测结果的准确度及泛化能力,选择二维三次插值的方式将原训练集 19 组样本数据增加至 100 组样本数据,重新构造训练集样本数据。

```
N = size(trainx0,2);              %获取当前训练集样本数据个数
X = [trainx0; trainy0];           %将特征输入量与期望输出量合并,构成完整训练集样本数据
[xx0, yy0] = meshgrid(1:N, 1:6);
[xx1,yy1] = meshgrid(linspace(1,N,100), 1:6); %构造网格矩阵,增加训练集样本个数至100
XX = interp2(xx0, yy0, X, xx1, yy1,'cubic'); %选择二维三次插值方式扩充训练集样本数据量
trainx = XX(1:5, :);              %构造训练集样本特征输入量数据
trainy = XX(6, :);                %构造训练集样本期望输出量数据
```

2) 搭建 RBF 神经网络并且训练网络

使用 MATLAB 神经网络工具箱的 newrb 函数初始化 RBF 神经网络,并设置相关的训练参数以创建地下水位预测模型。设置网络的训练目标误差为 1e-8,将径向基函数的扩展系数设置为 1,根据训练集的样本个数设置最大隐含层节点个数为 100,在训练过程中以“1”为单位,逐步增加神经元的个数直到训练误差低于目标误差。

```
net = newrb(train_x,train_t,1e-8,1,100,1);                %创建 RBF 神经网络
```

3) 测试集数据样本验证 RBF 神经网络模型预测效果

通过测试集样本数据验证基于 RBF 神经网络的地下水位预测模型的预测准确度,计算神经网络输出预测结果与数据集期望输出结果之间的相对误差,以及全部测试集样本数据的平均相对误差和最大相对误差。通过计算预测结果与期望输出结果之间的相对误差,可以对模型的泛化能力进行评价,并在此基础上,进一步研究和改善 RBF 神经网络参数,相对误差越小,则表明 RBF 神经网络模型预测结果越接近期望输出结果,RBF 神经网络预测模型泛化能力越强。将测试集样本数据神经网络输出预测结果与数据集期望输出结果显示在同一坐标系下,可以更直观地看出预测结果与期望结果之间的相对误差。

```
yy = net(testx);
e = (testy - yy)./testy;          %计算网络输出预测结果与期望输出结果之间的相对误差
fprintf('相对误差:\n');
fprintf('%f', e);
fprintf('\n\n');
m = mean(abs(e));                 %计算测试集样本数据的平均相对误差
fprintf('平均相对误差:\n %f\n', m);
ma = max(abs(e));                 %计算测试集样本数据的最大相对误差
fprintf('最大相对误差:\n %f\n', ma);
figure(1)                         %显示实际值与拟合值
plot(1:5, testy, 'bo-')
hold on
plot(1:5, yy, 'r*-')
title('地下水位测试结果')
legend('真实值', '预测值')
axis([1,5,0,8])
```

5. 实验结果分析

训练集输入样本数据插值前后对比如图 5.16 所示，训练集期望输出数据插值前后对比如图 5.17 所示。

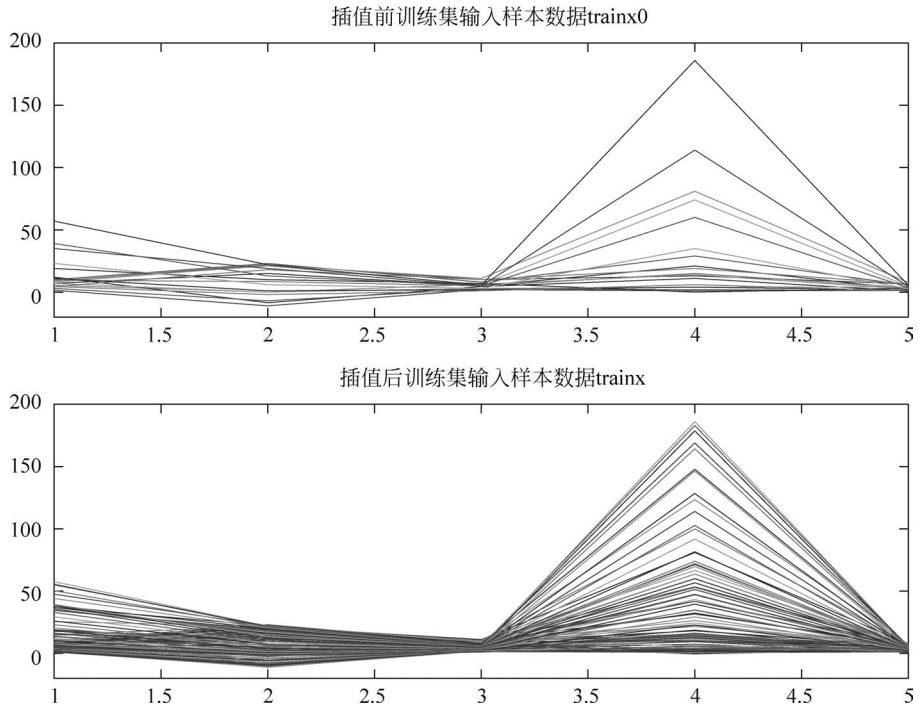

图 5.16　训练集输入样本数据插值前后对比

经过 RBF 神经网络地下水位预测模型训练后，测试集样本数据预测结果与期望输出结果之间的相对误差如表 5.6 所示，由表可知测试样本最大相对误差为 0.166 278。测试集预测结果与期望输出结果对比如图 5.18 所示。

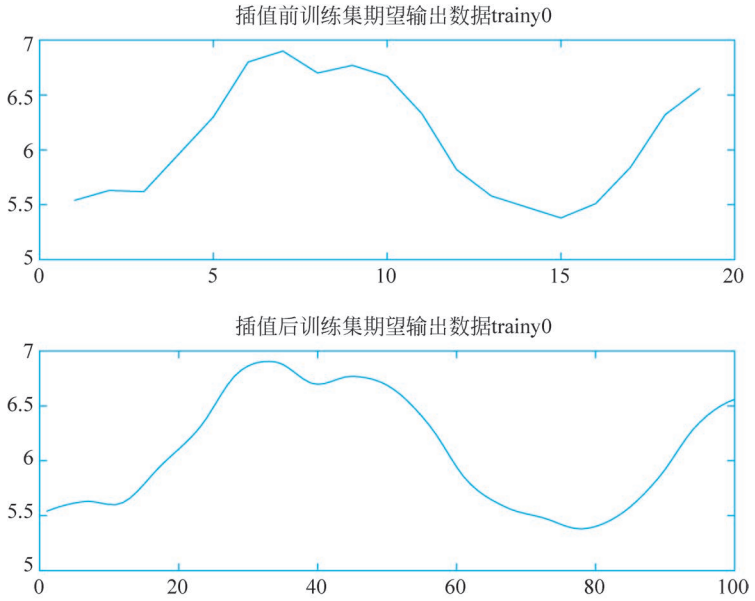

图 5.17 训练集期望输出数据插值前后对比

表 5.6 测试集样本数据相对误差

样 本 编 号	期望输出结果	RBF 神经网络预测结果	相 对 误 差
1	6.92	6.7161	0.029 470
2	6.97	6.7216	0.035 646
3	6.84	6.7068	0.019 474
4	6.5	6.7137	−0.032 881
5	5.75	6.7061	−0.166 278

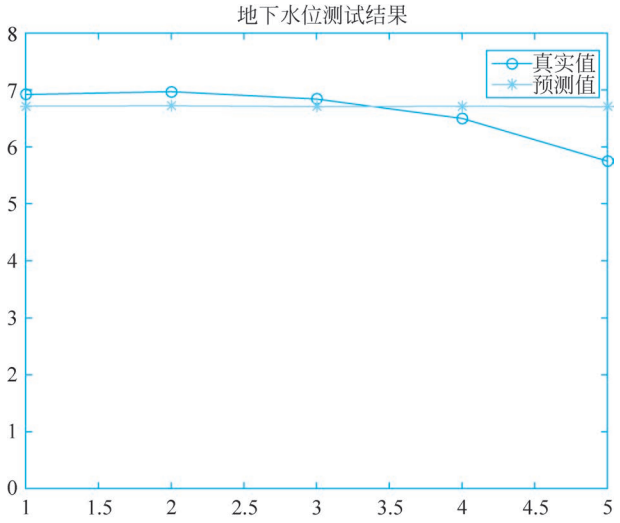

图 5.18 测试集预测结果与期望输出结果对比图

通过图 5.18 可以看出，测试集前 4 个样本数据的 RBF 神经网络地下水位预测结果相对误差较小，预测结果接近期望输出结果，RBF 神经网络地下水位预测模型数据拟合准确

程度较高。但是第 5 个样本数据的 RBF 神经网络地下水位预测结果相对误差较大，表明该 RBF 神经网络地下水位预测模型有待改进。

本章习题

1. 正则化 RBF 神经网络和广义 RBF 神经网络的主要区别是什么？
2. 广义 RBF 神经网络的径向基函数的选取方法有哪些？
3. RBF 神经网络与 BP 神经网络相比，有哪些优缺点？
4. 简述广义 RBF 神经网络的学习算法流程。
5. 本章 5.3.1 节基于 MATLAB 的 RBF 神经网络应用案例——数据拟合案例中，隐含层节点选取了 3 个，现选取 4 个隐含层节点，实现数据拟合，并将数据拟合效果与 5.3.1 节应用案例的拟合效果进行比较。
6. 本章例 5.1 “异或”问题采用正则化 RBF 神经网络进行解决，其中隐含层节点数选取了 2 个，其数据中心选为输入的 2 个模式 $\boldsymbol{X}^1=\begin{pmatrix}1\\1\end{pmatrix}$、$\boldsymbol{X}^2=\begin{pmatrix}0\\0\end{pmatrix}$。现选取 4 个隐含层节点，其数据中心选为输入的 4 个模式 $\boldsymbol{X}^1=\begin{pmatrix}1\\1\end{pmatrix}$、$\boldsymbol{X}^2=\begin{pmatrix}0\\0\end{pmatrix}$、$\boldsymbol{X}^3=\begin{pmatrix}1\\0\end{pmatrix}$、$\boldsymbol{X}^4=\begin{pmatrix}0\\1\end{pmatrix}$，设计正则化 RBF 神经网络解决“异或”问题。