

# 全国第七届研究生数学建模竞赛



## 题 目 神经元形态分类与识别的数学建模

### 摘 要：

本文针对神经元的形态分类与识别问题，运用决策树、神经网络、支持向量机（SVM）等分类方法对问题进行求解。

对于问题 1，本文先采用单一决策树模型将神经元划分为 7 类，通过交叉验证，分类的准确率为 84.3%。然后，采用多决策树模型对神经元进行分类，第一步分类的准确率为 90.2%，第二步分类的准确率为 90.9%。接着，采用神经网络模型将神经元分别划分为 5 类、7 类，分类的准确率分别为 98.04%、88.24%。最后，采用 SVM 模型将神经元分别划分为 5 类、7 类，分类的准确率分别为 98.04%、86.27%。同时得到刻画附录 C 中 5 类神经元的几何特征为胞体表面积，主干数目，分支数目，分枝级数和高度。

对于问题 2，在对测试集进行扩展后，本文先后采用神经网络模型和 SVM 模型对附录 B 中的未知神经元利用概率输出来作类型识别及新类探测。最后得到一致的识别结果，1~3 号为椎体神经元，5~6 号为普肯野神经元，7~9 为新类神经元，4、10~12、19~20 号为运动神经元，13~14、17~18 号为感觉神经元，15~16 为中间神经元。

对于问题 3，本文利用神经网络和 SVM 概率输出来作类型识别及新类探测。

对于问题 4，本文采用在训练集中进一步细分的方法来验证神经网络模型以及 SVM 解决该问题的有效性。

对于问题 5，本文先通过对树的生长进行模拟，然后利用 L-system 的建模方法实现对神经元生长的模拟，从而预测出神经元的生长变化。

**关键字：**神经元 形态分类 决策树 神经网络 支持向量机 分形

参赛队号 10248012

队员姓名 潘玮 周徐宁 仇伟

参赛密码 \_\_\_\_\_  
(由组委会填写)

中山大学承办

# 目录

一、问题重述.....	3
二、模型假设.....	3
三、符号说明.....	3
四、名词解释.....	4
五、问题的分析.....	4
5.1 问题 1 分析.....	5
5.2 问题 2 分析.....	5
5.3 问题 3 分析.....	5
5.4 问题 4 分析.....	5
5.5 问题 5 分析.....	5
六、模型一：决策树模型.....	5
6.1 决策树(C4.5)介绍 .....	5
6.2 问题 1 求解：单一决策树.....	6
6.4 问题 1 求解：多决策树.....	8
6.5 总结.....	10
七、模型二：神经网络模型.....	10
7.1 神经网络（ANN）介绍 .....	10
7.2 BP 神经网络模型的建立.....	11
7.4 基于神经网络概率分布输出的新类探测.....	13
7.5 问题 3：命名建议.....	19
八、模型三：支持向量机.....	20
8.1 支持向量机 SVM 介绍 .....	20
8.2 问题 1 求解：SVM 分类.....	20
8.3 问题 2：类型识别及新类探测.....	22
8.4 问题三：命名建议.....	28
8.5 问题四：不同种类动物同种类别之间神经元区别.....	28
九、模型四：神经元生长模型.....	29
9.1 L-System 建模 .....	29
9.2 字符串替换.....	29
9.3 龟形建模.....	29
9.4 基于 L-System 的神经元生长的实现.....	31
十、模型的评价.....	32
参考文献.....	33
附录.....	34

## 一、问题重述

大脑是生物体内结构和功能最复杂的组织，其中包含上千亿个神经细胞（神经元）。作为大脑构造的基本单位，神经元的结构和功能包含很多因素，其中神经元的几何形态特征和电学物理特性是两个重要方面。其中电学特性包含神经元不同的电位发放模式；几何形态特征主要包括神经元的空间构象，具体包含接受信息的树突，处理信息的胞体和传出信息的轴突三部分结构。由于树突，轴突的生长变化，神经元的几何形态千变万化。如何识别区分不同类别的神经元，已经成为人类脑计划中的一个重要项目。在生物解剖上，主要通过几何形态和电位发放两个因素来区别神经元。

本问题只考虑神经元的几何形态，研究如何利用神经元的空间几何特征，通过数学建模给出神经元的一个空间形态分类方法，将神经元根据几何形态比较准确地分类识别。

问题 1、利用附录 A 和附录 C 样本神经元的空间几何数据，找出用于刻画附录 C 中 5 类神经元的几何特征（中间神经元可以又细分 3 类），并给出一种神经元形态分类方法。

问题 2、利用神经元形态分类方法，将附录 B 中的 20 个神经元进行分类，并进行新类探测，从而决定是否引入新的神经元类别。

问题 3、设计一种神经元形态分类方法，将所有神经元按几何特征进行分类，并给出神经元的命名方式。

问题 4、利用神经元形态分类方法，判别在不同动物神经系统中同一类神经元的形态特征。

问题 5、预测神经元形态随时间的变化，比较变化前后的几何形态特征。

## 二、模型假设

1. 假设不同动物的同类神经元的形态特征之间差异显著小于不同类神经元的形态特征之间差异。
2. 假设神经元根据其几何形态特征可分。
3. 假设用于分类的样本数据数目（训练集）足够大，能反应出分类标准。
4. 假设神经元生长受环境影响

## 三、符号说明

符号	符号说明
Soma_Surface	胞体表面积
Number_of_Stems	主干数目：指从神经元胞体生发而出的主干的数目
Number_of_Bifs	分叉数目：神经元中分叉点的数目
Number_of_Branch	分支数目：指从神经元主干生发而出的所有分支的数目
Width	宽度
Height	高度
Depth	深度
Diameter	直径：神经元的突起是由粗变细的椎体，抽象为一个圆柱而得到的直径

Length	长度：指神经元的突起末端中，两个路径距离最远的末端之间的路径距离
Surface	表面积：指神经元的表面积
Volume	体积：指神经元本身的体积
EucDistance	欧氏距离：神经元到距离它最近的特征点的物理距离
PathDistance	路径距离：指从神经元延着神经网络到达最近特征点的距离
BranchOrder	分叉级数：指一个神经元中，分叉数目最多的主干上的分叉点数
Contraction	压缩比：欧式距离与路径距离的比值
Fragmentatio	破碎程度：生物染色时不完全的程度
PartAsymme	非对称化
RallRatio	罗尔比率：树突出现分叉的前后两端，其直径之间的关系
BifAngLocal	局部分叉角：距离胞体最近的分叉的角度
BifAngRemo	远处分叉角：距离胞体最远的分叉的角度
motoneu	运动神经元，简写 m
purkneu	普肯野神经元，简写 pu
pyraneu	锥体神经元，简写 py
biponeu	双极中间神经元，简写 i_b
triponeu	三极中间神经元，简写 i_t
multineu	多极中间神经元，简写 i_m
interneu	中间神经元，简写 i，包括双极、三极、多极三类
senneu	感觉神经元，简写 s

#### 四、名词解释

名词	名词含义
原始数据集	指由附录 A 和附录 C 中样本神经元的空间几何数据构成的神经元空间形态特征集合，共有 51 条记录
扩展数据集	指为了提高假设 3 的可靠性，对原始数据集按类别进行等比例扩展数据所得到的空间形态特征集合，共包含 130 条记录
测试集 T	指为了评价模型，共扩展另外 15 条记录的集合，其中中间神经元 5 条，运动神经元 2 条，普肯野神经元 2 条，锥体神经元 2 条，感觉神经元 2 条，Calretinin 神经元 2 条。
测试集 B	指附录 B 中的空间数据构成的空间形态特征集合，共有 20 条记录

注：所有扩展的数据均来自 Neuronmorpho.prg 网站[1]，空间形态特征通过 L-measure 软件计算得到[2]。

#### 五、问题的分析

神经元空间几何形态的研究是人类脑计划中的一个重要项目。由于神经元的形态复杂多样，神经元的识别分类问题至今仍没有解决。生物解剖上主要通过几

何形态和电位发放两个因素来区别神经元。神经元的形态丰富多样，其形态特征的提取十分困难。根据文献[2]的特征描述和 Neuronmorpho.prg 的特征数据分类，选用神经元的胞体表面积，主干数目，分叉数目，宽度，深度，直径，长度，表面积，体积，欧氏距离，路径距离，分叉级数，压缩比，破碎程度，非对称化，罗尔比率，局部分叉角，远处分叉角等 20 个参数来刻画神经元的形态特征。本文将从神经元的形态和物理特征入手，研究神经元的分类和识别问题。

分类的目的是构造一个分类函数或分类模型（分类器），将数据库中的数据项映射到某一个给定类别中。分类问题的数学描述为，给定数据集合  $T = \{t_1, t_2 \cdots t_n\}$ ， $t_i \in T$  称之为样本，类集合  $C = \{c_1, c_2 \cdots c_m\}$ 。分类问题定义为从

数据集合到类集合的映射  $f: T \rightarrow C$ ，即数据集合中的样本  $t_i$  分配到某个类  $c_j$  中，

有  $c_j = \{t_i | f(t_i) = c_j, 1 \leq i \leq n, \text{且 } t_i \in T\}$ 。

分类问题有很多解决方法和技术[3]，本文采用决策树、神经网络、支持向量机来进行分类。

### 5.1 问题 1 分析

利用分类方法，先通过对原始数据集进行分类，得到分类的标准与规则，利用刻画此分类标准的参数作为 C 类中神经元的几何特征。

### 5.2 问题 2 分析

利用神经元形态分类方法，将测试集 B 中的 20 条记录输入分类器进行分类，同时进行新类探测，当分类为指标不在规定范围内，则决定引入新的神经元类别。

### 5.3 问题 3 分析

利用神经元形态分类方法，将神经元的形态特征输入分类器进行分类，当分类指标不在规定范围内，则将此神经元归入新类。当新类种数达到一定数目时，再将此新类进行分类，并按照分类的路径进行神经元命名。

### 5.4 问题 4 分析

利用神经元形态分类方法，将在不同动物神经系统中同一类神经元的形态特征输入分类器中，可得到输出概率，根据输出概率的比较来确定是否属于同一类。

### 5.5 问题 5 分析

根据自然界生物的生长特性，模拟神经元生长随时间的变化。计算变化前后的神经元形态特征，输入分类器进行分类。

## 六、模型一：决策树模型

### 6.1 决策树(C4.5)介绍

决策树(Decision Tree)是一种常用的数据分类方法，它具有类似流程图的树结构，其中每个内部节点表示在一个属性上的测试，每个分枝代表一个测试输出，而每个树叶节点代表类或类分布。

决策树归纳的基本算法是贪心算法，它以自顶向下递归的各个击破方式构造决策树。决策树在构造过程中需要选择内部节点，也就是对某个属性做测试。一般采用信息增益(Information Gain)度量或类似指标选择测试。通过这种方式选出的属性使得对结果划分中的样本分类所需的信息量最小，并反映划分的最小随机性或“不纯度”[4]。

本文采用一种常见的决策树构造算法 C4.5，它利用信息增益率(Information Gain Ratio)作为节点选取标准。设  $S$  是  $s$  个数据样本的集合。假定类标号属性具有  $m$  个不同值，定义  $m$  个不同类  $C_i (i=1,2,\dots,m)$ 。设  $s_i$  是类  $C_i$  中的样本数。对一个给定的样本分类所需的期望信息由下式给出：

$$I(s_1, s_2, \dots, s_m) = - \sum_{i=1}^m p_i \log_2(p_i) \quad (1)$$

其中， $p_i$  是任意样本属于  $C_i$  的概率，并用  $s_i/s$  估计。

设属性  $A$  具有  $v$  个不同值  $\{a_1, a_2, \dots, a_v\}$ 。可以用属性  $A$  将  $S$  划分为  $v$  个子集  $\{S_1, S_2, \dots, S_v\}$ ；其中， $S_j$  包含  $S$  中这样一些样本，它们在  $A$  上具有值  $a_j$ 。如果  $A$  选做测试集属性（即最好的分裂属性），则这些子集对应于由包含集合  $S$  的节点生长出来的分枝。设  $s_{ij}$  是子集  $S_j$  中类  $C_i$  的样本数。根据由  $A$  划分成子集的熵（Entropy）或期望信息由下式给出：

$$E(A) = \sum_{j=1}^v \frac{s_{1j} + \dots + s_{mj}}{s} I(s_{1j}, \dots, s_{mj}) \quad (2)$$

项  $\frac{s_{1j} + \dots + s_{mj}}{s}$  充当第  $j$  个子集的权，并且等于子集（即  $A$  值为  $a_j$ ）中的样本个数除以  $S$  中的样本总数。熵值越小，子集划分的纯度越高。

在  $A$  上分枝将获得的编码信息是

$$Gain(A) = I(s_1, s_2, \dots, s_m) - E(A) \quad (3)$$

信息增益率定义为

$$GainRatio = \frac{Gain(S, A)}{SplitInfo(S, A)} \quad (4)$$

其中

$$SplitInfo(S, A) = - \sum_{i=1}^c \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|} \quad (5)$$

## 6.2 问题 1 求解：单一决策树

为了刻画附录 C 中涉及的 5 类神经元（中间神经元又可以细分 3 类）的几何特征，本文通过单一决策树的模型来求解。

先将中间神经元的三个分类（两极中间神经元、三级中间神经元、多级中间神经元）作为独立分类，与其他类别神经元（运动神经元、普肯野神经元、锥体神经元）放在同一个分类层次中。通过寻找一种分类标准能够直接区别出这 7 个类，以获得分类标准，从而达到获取 7 类神经元的几何特征。

利用 6.1 节提到的 C4.5 算法，使用 Java 语言并利用 weka 工具中的 J18 工具

包编程，针对原始数据集，得到如图 1 所示的决策树。

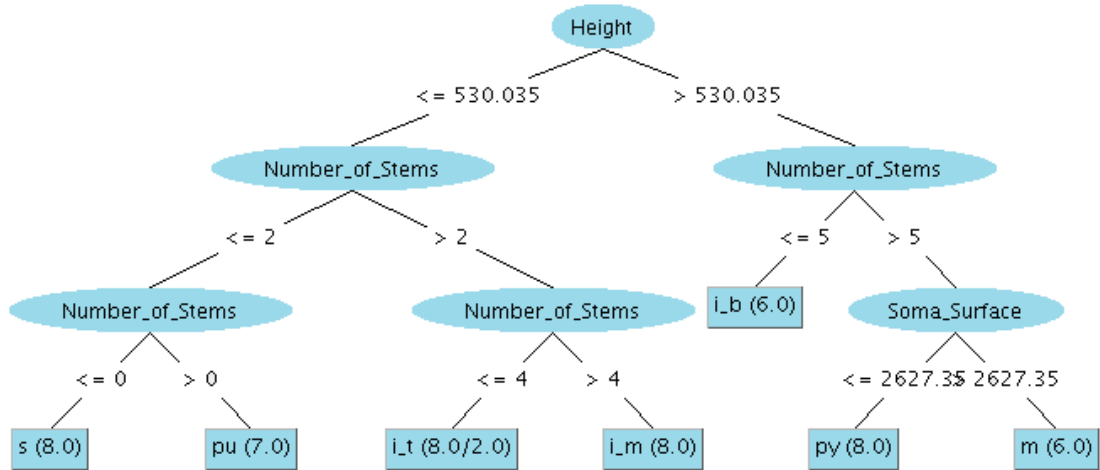


图 1 问题 1 求解的单一决策树

根据以上决策树的建立过程，可知提取到神经元的主要分类指标为：高度，主干数目，胞体表面积。

由于原始数据集的样本容量偏小，因此本文采用交叉验证的方式来评价该模型。本文中采用 10 折交叉验证，得到的混淆矩阵如表 1 所示。

表 1 测试结果

样本 归类\	biponeu	multineu	triponeu	motoneu	purkneu	pyraneu	senneu	准确率
biponeu	4	0	1	0	1	0	0	0.667
multineu	0	8	2	0	0	0	0	0.889
triponeu	1	1	4	0	0	0	0	0.571
motoneu	0	0	0	5	0	1	0	1.000
purkneu	1	0	0	0	6	0	0	0.857
pyraneu	0	0	0	0	0	8	0	0.889
senneu	0	0	0	0	0	0	8	1.000

从上表可知，测试的总体准确率为  $43/51=84.3137\%$ 。

因此，可以得出结论，附录 C 中 7 类神经元的几何特征可以通过以上三个分类指标来刻画，即高度，主干数目和胞体表面积。抽取特征定义如表 2 所示。

表 2 单一决策树抽取特征定义

神经元类别	抽取特征
双极中间神经元	$Height > 530.035 \ \& \ Number\_of\_Stems \leq 5$
多极中间神经元	$Height \leq 530.035 \ \& \ Number\_of\_Stems > 4$
三极中间神经元	$Height \leq 530.035 \ \& \ 2 < Number\_of\_Stems \leq 4$

运动神经元	$Height > 530.035 \& Number\_of\_Stems > 5 \& Soma\_Surface > 2627.35$
普肯野神经元	$Height \leq 530.035 \& Number\_of\_Stems > 0$
锥体神经元	$Height > 530.035 \& Number\_of\_Stems > 5 \& Soma\_Surface \leq 2627.35$
感觉神经元	$Height \leq 530.035 \& Number\_of\_Stems \leq 0$

为了进一步验证模型的有效性，我们在扩展数据集上也做类似的交叉验证，得到的准确率为  $104/130=80\%$ ，与原始数据集相比，准确率有一定下降，但并不显著。

#### 6.4 问题 1 求解：多决策树

由于单一决策树过于简单，有时候复杂分类问题并不是通过这样的规则就可分的，所以导致了单一决策树在神经元分类问题上的准确率并不高。另外注意到混淆矩阵当中属于中间神经元的三类神经元准确率明显低于其他 4 个大类，并且有超过一半错误都是这三类之间的分类错误。因此，可以质疑这是忽略了这 7 个类存在的层次关系所致。基于这个发现，本文提出分步建立决策树充分利用 7 类神经元之间的分类体系关系。

根据问题一可知，附录 C 中神经元的类别存在着如图 2 所示的关系。

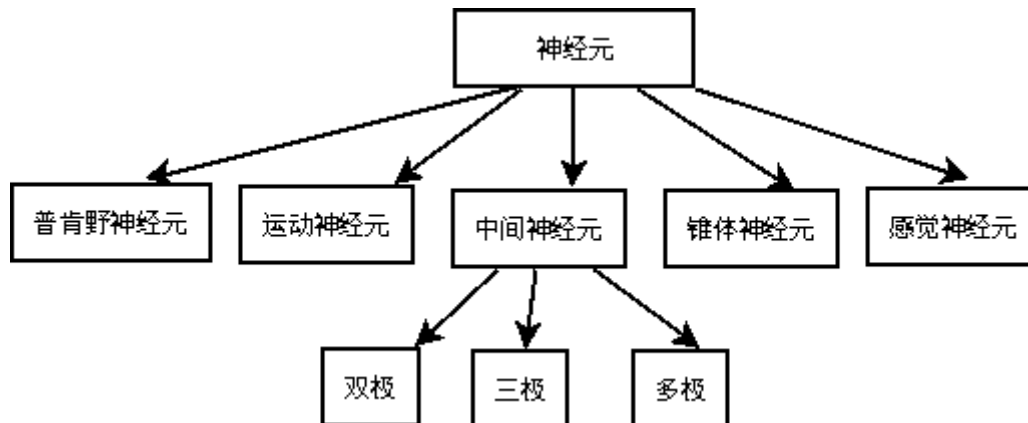


图 2 神经元类别的关系

因此，参照这个神经元类别之间的关系，建立一个决策树先对处于类别关系第一层次的五类神经元进行分类，然后再建立另一个决策树对中间神经元进行分类。

利用 6.1 节提到的 C4.5 算法，使用 Java 语言并利用 weka 工具中的 J48 工具包编程，针对原始数据集，得到如图 3、图 4 所示的决策树。



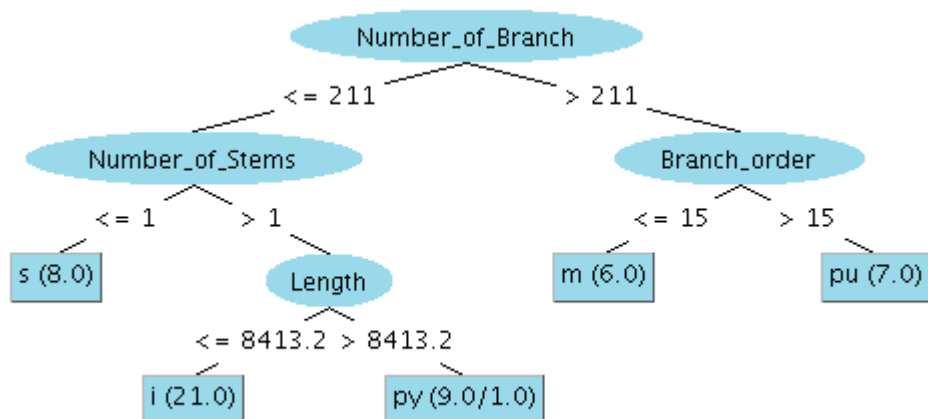


图 3 问题 1 求解的第一层决策树

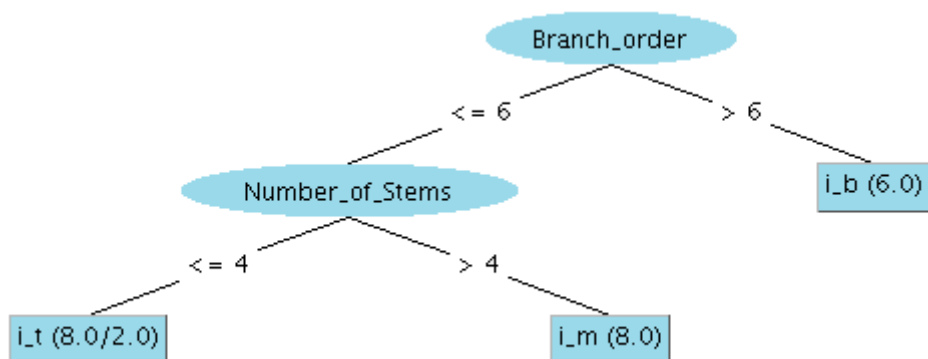


图 4 问题 1 求解的第二层决策树

利用与 6.2 节类似的 10 折交叉验证来评估得到的模型，得到第一层决策树的混淆矩阵如表 3 所示。

表 3 测试结果

样本 种类	interneu	motoneu	purkneu	pyraneu	senneu
interneu	18	0	2	2	0
motoneu	0	6	0	0	0
purkneu	0	1	6	0	0
pyraneu	0	0	0	8	0
senneu	0	0	0	0	8

第二层决策树的混淆矩阵如表 4 所示。

表 4 测试结果

样本 种类	biponeu	multineu	triponeu
biponeu	6	0	0
multineu	0	9	1
triponeu	0	1	5

根据表 3、表 4 可知，利用决策树进行分类，第一步分类的准确率约为 90.2%，第二步分类的准确率约为 90.9%。

在扩展数据集上的交叉验证表明，采用决策树算法构造出的模型能在第一步获得约 91.5%的准确率，第二步获得约 82.2%的准确率。

## 6.5 总结

本文利用决策树模型（C4.5）算法作出了解决问题的第一个尝试。从上文的描述中可以看到，决策树模型是一个简洁有效的分类模型，针对我们需要解决的神经元按照空间特征分类问题，它在原始测试集以及扩充测试集上按照神经元空间形态的分类性能达到了均达到了 80%以上。同时决策树模型还非常自然地在 20 个形态特征中抽取出了对分类起到关键作用的形态特征作为 5 类神经元的几何特征，并显示的给出了人们可以理解，掌握的分类依据。

根据构造出的决策树结构，专家可以据此拟定命名规则。例如图 3 中的中间神经元我们可以参照决策树结构按照路径给它拟名“少分支多主干短神经元”。类似地，如果有全部神经元的分类决策树，我们也可以用类似的方法给它命名。尽管这样的命名方法略显臃肿，但它却能在名字中就附带出它的形态特征。

依据决策树建立的模型也有着不可避免的缺陷。首先，由于神经元空间形态分类问题固有的复杂性，简单的决策树分类方法很难达到非常高准确率。同时这种方法缺少对预测类标置信度的估计能力，因此很难直接利用决策树本身的信息来对新出现的类别进行探测。同时该模型对分到同一个叶子节点下的记录不具备区分能力。因此该模型不能直接解决问题 2（虽然它能够对附录 B 中的数据进行分类，但是不具备发现新类的能力，因此我们这里不采用它对问题 2 进行回答）、问题 4，我们将在下面的章节中予以回答。

至此，我们采用决策数模型回答了问题 1，问题 3，同时指出它也具备部分解决问题 2 的能力。

## 七、模型二：神经网络模型

### 7.1 神经网络（ANN）介绍

神经网络是由大量的简单的处理单元广泛的互相连接而形成的复杂网络系统，它反映了人脑功能的许多基本特征。神经网络具有大规模并行、分布式存储和处理、自组织和自学习能力。

最基本的神经网络如图 5 所示，它只包含一个神经单元，又称为单层感知器。

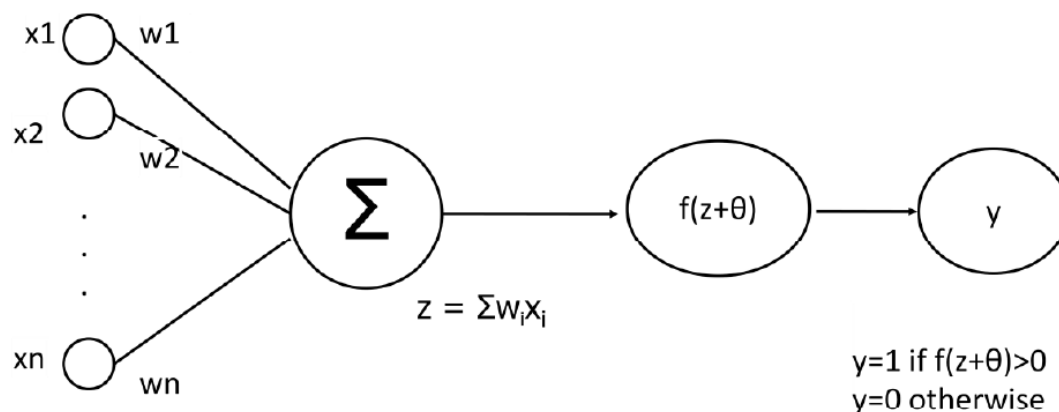


图 5 单层感知器结构框图

其中  $x$  为外界信息的输入，各种信息通过突触上不同的权重进行加权求和，即以  $z = \sum_i w_i x_i$  作为神经元的激励，而函数  $f(x + \theta)$  称为激活函数，作为神经元的输出函数。

神经网络广泛应用于分类识别方法中，通过在训练过程中有监督的学习，不断地通过学习，改变突触的权值，最终完成一个对识别系统的模拟。由于本问题涉及的识别类别数目较多，因此需要选择具有多层结构神经网络，例如如图 6 所示的两层 BP 神经网络。

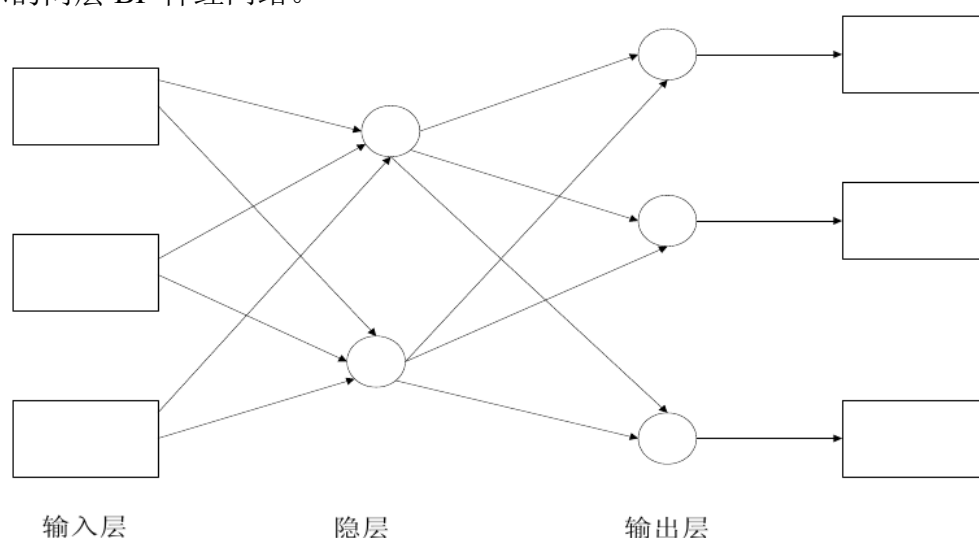


图 6 两层神经网络的框架图

多层神经网络中的每个神经单元模型包括一个非线性激活函数，正是由于多层神经网络中隐层神经元的存在，使得网络可以从输入模式向量中提取特征，使网络学习复杂的任务。

## 7.2 BP 神经网络模型的建立

我们采用两层 BP 神经网络建模，选取 16 个隐藏节点，为了防止过度拟合，我们在训练过程中采用 10 折交叉验证的方法，针对原始数据集，采用 5 类神经元进行分类，得到混淆矩阵如表 5 所示。

表 5 测试结果

样本 种类	interneu	motoneu	purkneu	pyraneu	senneu
interneu	21	0	0	0	1
motoneu	0	6	0	0	0
purkneu	0	0	7	0	0
pyraneu	0	0	0	8	0
senneu	0	0	0	0	8

从上表可知，测试的总体准确率为 98.04%。

对原始数据集，采用 7 类神经元得到混淆矩阵如表 6 所示。

表 6 测试结果

样本 种类	biponeu	multineu	triponeu	motoneu	purkneu	pyraneu	senneu
biponeu	6	0	0	0	0	0	0
multineu	0	7	3	0	0	0	0
triponeu	0	3	3	0	0	0	0
motoneu	0	0	0	6	0	0	0
purkneu	0	0	0	0	7	0	0
pyraneu	0	0	0	0	0	8	0
senneu	0	0	0	0	0	0	8

从上表可知，测试的总体准确率为 88.24%。

由此可以看出，BP 神经网络对于 5 类神经元有着很好的分类能力，对 7 类神经元（包含了中间神经元的 3 个子类）也有着较好的分类能力，并且错误集中在中间神经元的三个子类上，这样的结果也和模型一中的观察一致。

进一步在扩展数据集上运用 10 折交叉验证评估上述模型的有效性。

采用 5 类神经元进行分类，得到混淆矩阵如表 7 所示。

表 7 测试结果

样本 种类	interneu	motoneu	purkneu	pyraneu	senneu
interneu	43	0	0	2	0
motoneu	0	18	0	3	0
purkneu	0	0	11	0	0
pyraneu	1	2	0	23	1
senneu	0	0	0	0	26

从上表可知，测试的总体准确率为 93.08%。

采用 7 类神经元进行分类，得到的混淆矩阵如表 8 所示

表 8 测试结果

样本 种类	biponeu	multineu	triponeu	motoneu	purkneu	pyraneu	senneu
biponeu	8	0	0	0	0	0	0
multineu	0	20	3	0	0	0	0
triponeu	0	0	13	0	0	1	0
motoneu	0	1	0	18	0	2	0
purkneu	0	1	0	0	10	0	0
pyraneu	0	0	0	1	0	26	0
senneu	0	0	1	0	0	0	25

从上表可知，测试的总体准确率为 92.31%。

#### 7.4 基于神经网络概率分布输出的新类探测

为了在预测未知种类的神经元类别的同时，判断出一些异常数据（这些异常数据往往代表他们不在现有的分类体系中）。本文通过对未知种类进行概率分布输出来发现这种异常，以达到新类探测的目的，神经网络的概率输出我们主要参考了文献[5]。

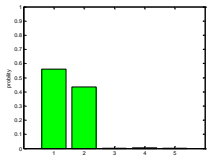
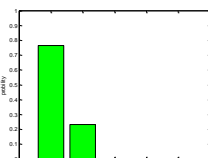
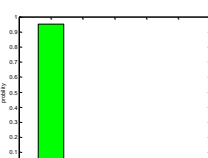
为了在预测未知种类的神经元类别的同时，判断出一些异常数据（这些异常数据往往代表他们不在现有的分类体系中）。本文通过对未知种类进行概率分布输出来发现这种异常，以达到新类探测的目的。

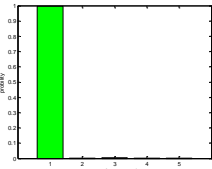
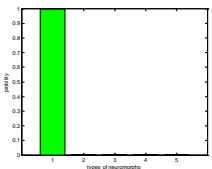
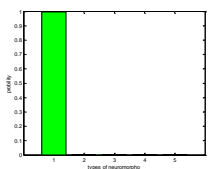
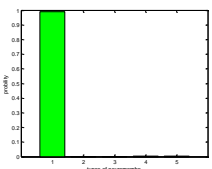
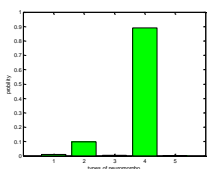
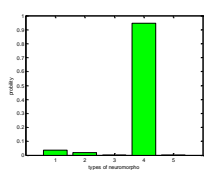
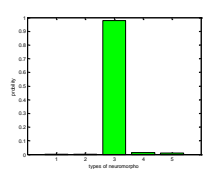
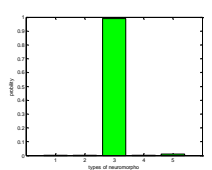
首先通过在原始数据集上训练 BP 网络。原始数据集上的记录通过 BP 网络预测后能够得到它在标签集上的概率分布，然后选取概率最大的类标作为该记录的类标。

假设获得的分类器对类标集合  $\{c_1, c_2, \dots, c_m\}$  都有一个置信度输出  $p_1, p_2, \dots, p_m$ 。选取概率最大者的类型为该神经元的类型，当概率最大值大于  $\alpha$  且最大值与次大值之差大于  $\beta$  时，则判定此神经元为新类神经元。其中  $\alpha, \beta$  为可调节的参数，用以控制对与新类的敏感性。对于  $\alpha, \beta$  的获取，我们可以采用在训练集上（或其他已经知道分类结果标签的测试集）根据所有正确分类的记录求出保证分类正确的  $\alpha, \beta$  的上确界。

为了检验上述模型的有效性，本文从 Neuronmorpho.prg 网站上获取了 15 条记录用来测试，其中中间神经元 5 条，运动神经元 2 条，普肯野神经元 2 条，锥体神经元 2 条，感觉神经元 2 条，另有两条不在以上 5 种当中的 Calretinin 神经元。将 15 条记录输入神经网络分类器，得到的输出概率如表 9 所示。

表 9 模型测试结果

神经元种类	输出概率图	类型识别	是否正确
Calretinin		异类	正确
Calretinin		异类	正确
中间神经元 1		中间神经元	正确

中间神经元 2		中间神经元	正确
中间神经元 3		中间神经元	正确
中间神经元 4		中间神经元	正确
中间神经元 5		中间神经元	正确
运动神经元 1		锥体神经元	错误
运动神经元 2		锥体神经元	错误
普肯野神经元 1		普肯野神经元	正确
普肯野神经元 2		普肯野神经元	正确

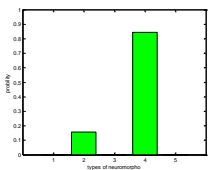
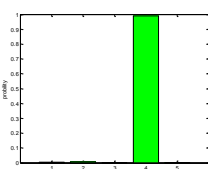
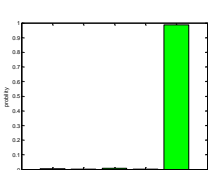
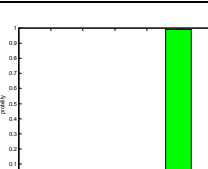
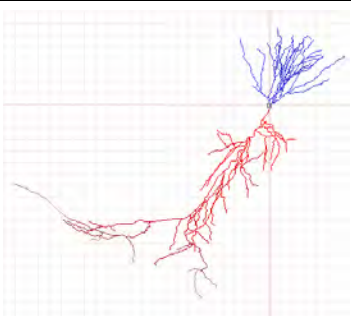
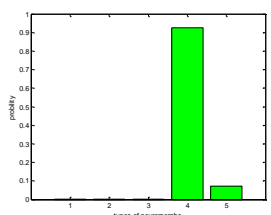
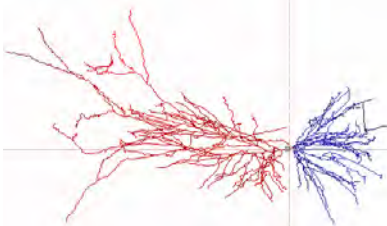
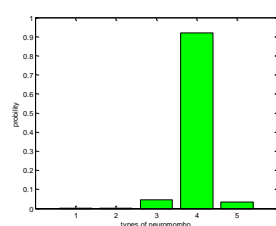
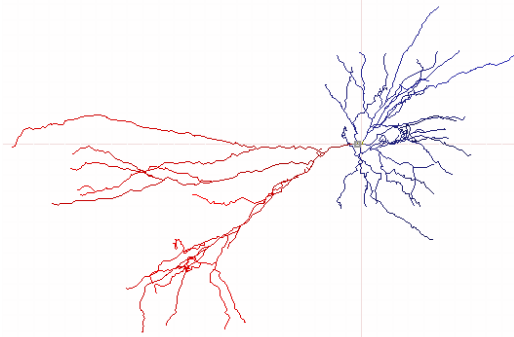
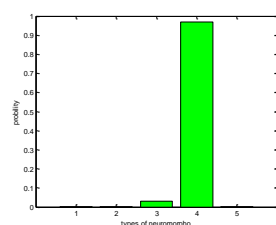

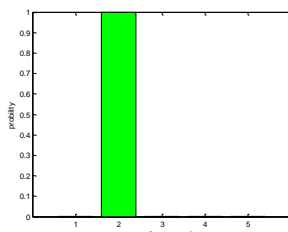
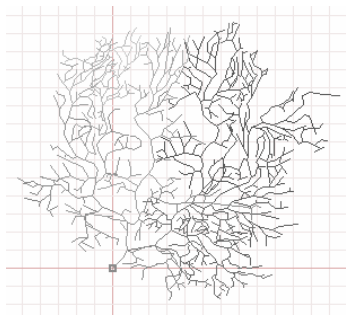
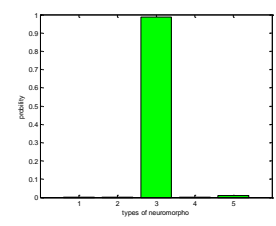

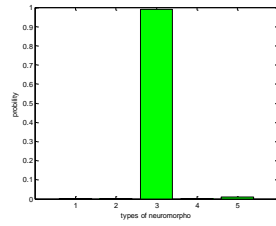
锥体神经元 1		锥体神经元	正确
锥体神经元 2		锥体神经元	正确
感觉神经元 1		感觉神经元	正确
感觉神经元 2		感觉神经元	正确

表 9 所示的输出概率图，水平轴的条形图依次代表中间神经元、运动神经元、普肯野神经元、锥体神经元、感觉神经元。从表 9 中可以看出，测试集 T 输入到神经网络分类器后，15 条记录共有 13 条识别正确，准确率约为 86.67%。从而，可以得出结论，神经网络分类器对神经元的形态识别十分有效。

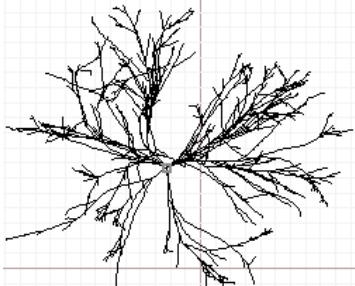
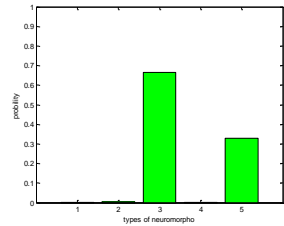
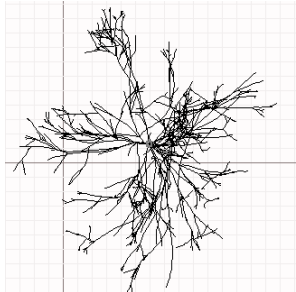
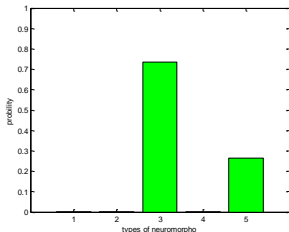
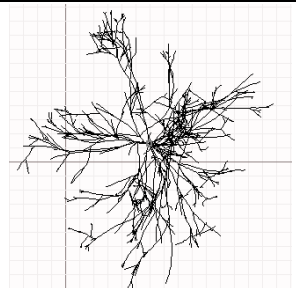
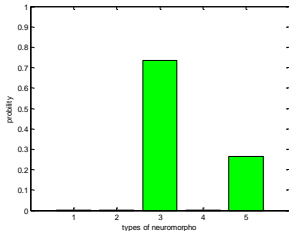
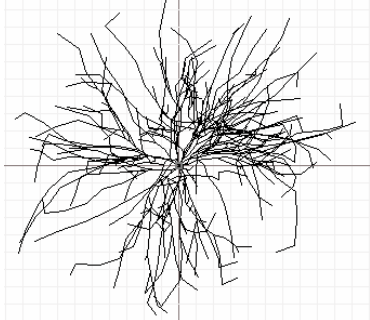
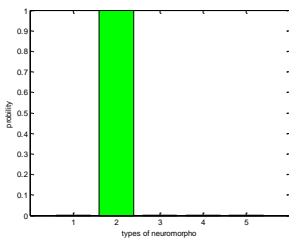
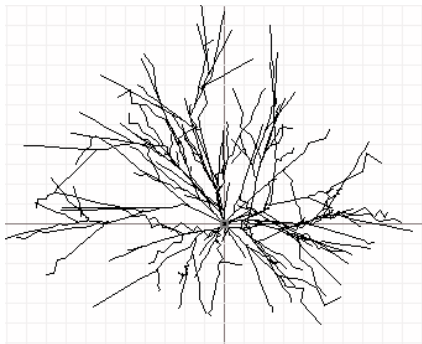
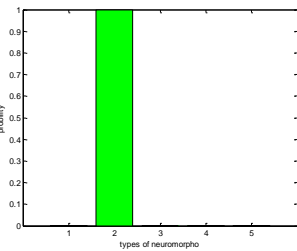
经过对神经网络分类器的有效性检验，将测试集 B 中的 20 条记录输入到分类器中，进行类型识别及新类探测，得到输出概率如表 10 所示。

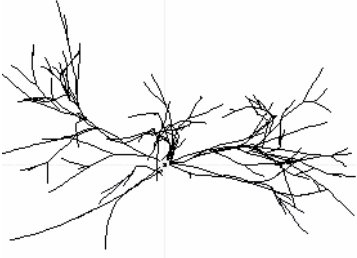
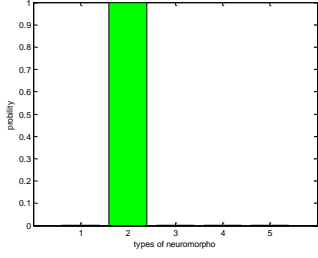

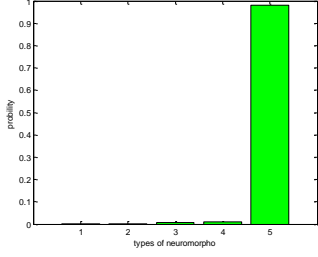
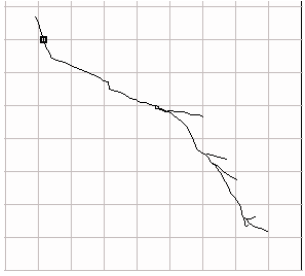
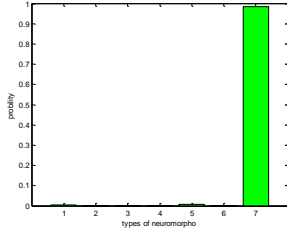
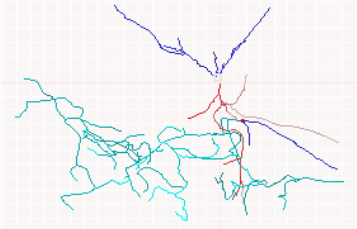
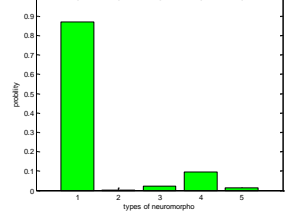
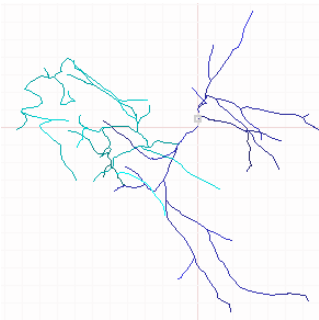
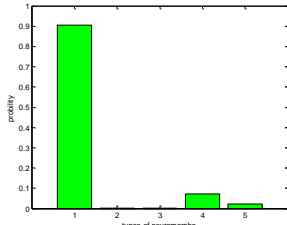

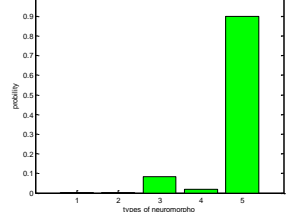
表 10 测试集 B 的类型识别及新类探测

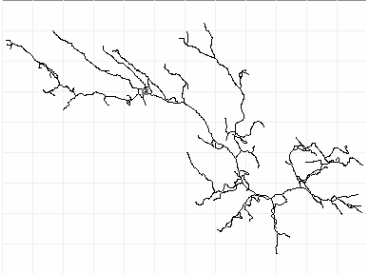
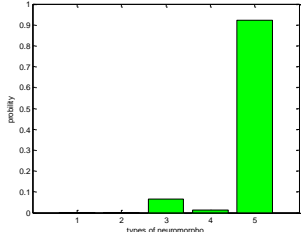
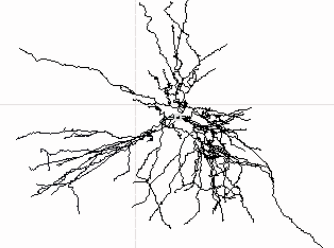
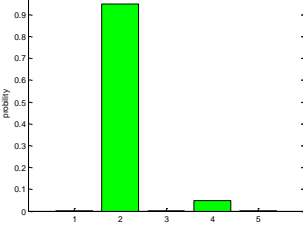
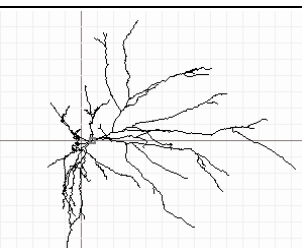
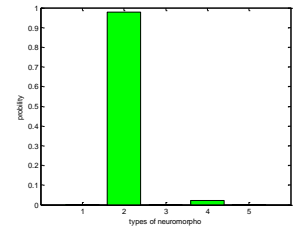
神经元编号	神经元二维图	输出概率图	类型识别
1			锥体神经元

2		 <table><tr><th>types of neuromorpho</th><th>probability</th></tr><tr><td>1</td><td>0.00</td></tr><tr><td>2</td><td>0.00</td></tr><tr><td>3</td><td>0.05</td></tr><tr><td>4</td><td>0.90</td></tr><tr><td>5</td><td>0.05</td></tr></table>	types of neuromorpho	probability	1	0.00	2	0.00	3	0.05	4	0.90	5	0.05	椎体神经元
types of neuromorpho	probability														
1	0.00														
2	0.00														
3	0.05														
4	0.90														
5	0.05														
3		 <table><tr><th>types of neuromorpho</th><th>probability</th></tr><tr><td>1</td><td>0.00</td></tr><tr><td>2</td><td>0.00</td></tr><tr><td>3</td><td>0.05</td></tr><tr><td>4</td><td>0.95</td></tr><tr><td>5</td><td>0.00</td></tr></table>	types of neuromorpho	probability	1	0.00	2	0.00	3	0.05	4	0.95	5	0.00	椎体神经元
types of neuromorpho	probability														
1	0.00														
2	0.00														
3	0.05														
4	0.95														
5	0.00														
4		 <table><tr><th>types of neuromorpho</th><th>probability</th></tr><tr><td>1</td><td>0.00</td></tr><tr><td>2</td><td>1.00</td></tr><tr><td>3</td><td>0.00</td></tr><tr><td>4</td><td>0.00</td></tr><tr><td>5</td><td>0.00</td></tr></table>	types of neuromorpho	probability	1	0.00	2	1.00	3	0.00	4	0.00	5	0.00	运动神经元
types of neuromorpho	probability														
1	0.00														
2	1.00														
3	0.00														
4	0.00														
5	0.00														
5		 <table><tr><th>types of neuromorpho</th><th>probability</th></tr><tr><td>1</td><td>0.00</td></tr><tr><td>2</td><td>0.00</td></tr><tr><td>3</td><td>1.00</td></tr><tr><td>4</td><td>0.00</td></tr><tr><td>5</td><td>0.00</td></tr></table>	types of neuromorpho	probability	1	0.00	2	0.00	3	1.00	4	0.00	5	0.00	普肯野神经元
types of neuromorpho	probability														
1	0.00														
2	0.00														
3	1.00														
4	0.00														
5	0.00														
6		 <table><tr><th>types of neuromorpho</th><th>probability</th></tr><tr><td>1</td><td>0.00</td></tr><tr><td>2</td><td>0.00</td></tr><tr><td>3</td><td>1.00</td></tr><tr><td>4</td><td>0.00</td></tr><tr><td>5</td><td>0.00</td></tr></table>	types of neuromorpho	probability	1	0.00	2	0.00	3	1.00	4	0.00	5	0.00	普肯野神经元
types of neuromorpho	probability														
1	0.00														
2	0.00														
3	1.00														
4	0.00														
5	0.00														



7			新类
8			新类
9			新类
10			运动神经元
11			运动神经元

12			运动神经元
13			感觉神经元
14			感觉神经元
15			中间神经元
16			中间神经元
17			感觉神经元

18			感觉神经元
19			运动神经元
20			运动神经元

注：神经元二维图由 Neuromantic 软件得到[6]。

根据表 10 可知，测试集 B 中共增加了 3 个新类，编号为 7，8，9。比较神经元 8，9 号的数据可知，8，9 号为同一条记录。通过观察编号 4 的二维图，可以看出，其二维图混乱，质疑其破碎程度比较高，有可能是由染色实验比较差所引起。因此，需通过对编号 4 的神经元进行具体分析，才能判断识别结果是否正确。

### 7.5 问题 3：命名建议

通过以上的讨论可以看出，在利用神经网络解决神经元根据空间形态特征分类时候，并没有显示地抽取出任何规则，分类标准实际上蕴含在了神经网络的结构当中。这就给我们像模型一中那样提出易于人理解的命名方式提出了困难。本文建议，在已有的 5 类神经元基础上构造完整的命名体系，先利用已有的几个关键类构造神经网络，然后使用 7.4 节处理大量的未知数据，每当发现新的类别，就提交给专家检验确认，并将专家确认后的数据添加到训练集中。最终我们将得到一个能区分所有已知神经元的神经网络模型，这样就可以根据输出他们类标的人工神经网络节点的序号对他们命名。

### 7.6 问题 4 求解：不同种类动物同种类别神经元之间的区别

根据本文的分类方法，只要能在训练数据中区别的表示出神经元的属于不同的动物种类，我们就能够训练出能够区分不同动物的同种神经元。

为了验证模型确实具有这样的能力，将原始数据集上的普肯野神经元细分为土拨鼠普肯野神经元和鼠普肯野神经元（并去除了一个原来属于 A 附录未知动物种类的普肯野神经元），共计 6 种 50 条记录。利用 10 折交叉验证得到的混淆矩阵如表 11 所示。

表 11 测试结果

样本 种类	interneu	motoneu	pigpurkneu	ratpurkneu	pyraneu	senneu
interneu	22	0	0	0	0	0
motoneu	0	6	0	0	0	0
pigpurkneu	0	0	2	1	0	0
ratpurkneu	0	0	0	3	0	0
pyraneu	0	0	0	0	8	0
senneu	0	0	0	0	0	8

从上表可知，测试的总体准确率为 98%。

从上面的数据块可以看出，神经网络模型具备拓展到识别不同种动物的同种神经元的能力。同时，我们可以看到仅有的一个错误恰出现在土拨鼠和鼠之间，这也说明了同种神经元即使在不同的动物上也具备了比较高的相似性，相比与其他种类神经元，它们之间要更难区分一些。

## 八、模型三：支持向量机

### 8.1 支持向量机 SVM 介绍

支持向量机 (SVM) [7]是由 Vapnik 等人提出的一种有监督的用于模式识别以及数据分析的方法。它基于结构风险最小化理论之上在特征空间中建构最优分割超平面，使得学习器得到全局最优化，并且在整个样本空间上的期望风险以某个概率满足一定上界。一些研究表明，SVM 常常具备其他分类器不具备的一些优点，例如在小样本表现较好、具有比较好的泛化能力。因此，特别适合应用于本文神经元的分类。

SVM 常常用来构造二类分类器，虽然 SVM 也能支持多分类问题。但是我们这里不仅希望 SVM 能够输出分类的类标，还希望能够得到对应类标的概率。因此，本文采用二类分类的带有概率输出的 SVM 分类器，概率输出主要参考了文献[8]。

### 8.2 问题 1 求解：SVM 分类

为了采用二类 SVM 分类器完成多类分类，我们需要训练多个一 vs 其他分类器。例如，将属于中间神经元记录的作为一类，不属于中间神经元的记录作为另外一类，这样训练出的带概率输出的二类分类器就能给出待预测记录属于中间神经元的概率。针对所有类别训练这样的分类器，就能够获得待预测记录属于每个类别的概率，我们通过比较这些概率来获得待预测记录的预测类别。

使用 Java 语言编程，并利用 libsvm[9]和 weka 工具包。经过实验比较我们采用 RBF 作为核函数。

在原始数据集上采用 10 折交叉验证。对 5 类神经元进行分类，得到的混淆矩阵如表 12 所示。

表 12 测试结果

样本 种类	interneu	motoneu	purkneu	pyraneu	senneu
interneu	22	0	0	0	0
motoneu	0	6	0	0	0

purkneu	0	0	7	0	0
pyraneu	1	0	0	7	0
senneu	0	0	0	0	8

从上表可知，测试的总体准确率为 98.04%。

对 7 类神经元进行分类，得到混淆矩阵如表 13 所示。

表 13 测试结果

样本 种类	biponeu	multineu	triponeu	motoneu	purkneu	pyraneu	senneu
biponeu	6	0	0	0	0	0	0
multineu	0	10	0	0	0	0	0
triponeu	0	6	0	0	0	0	0
motoneu	0	0	0	6	0	0	0
purkneu	0	0	0	0	7	0	0
pyraneu	0	0	0	0	0	8	0
senneu	0	0	1	0	0	0	7

从上表可知，测试的总体准确率为 86.27%。

由此可以看出，SVM 对于 5 类神经元有着很好的分类能力，对 7 类神经元（包含了中间神经元的 3 个子类）也有着较好的分类能力，并且错误集中在中间神经元的三个子类上，这也符合前文中提到的两个模型的结果，这说明了中间神经元的子类之间确实有高度的相似性。

进一步在扩展数据集上运用 10 交叉验证评估上述模型的有效性。

对 5 类神经元进行分类，得到混淆矩阵如表 14 所示。

表 14 测试结果

样本 种类	interneu	motoneu	purkneu	pyraneu	senneu
interneu	44	0	0	0	1
motoneu	0	18	0	3	0
purkneu	0	1	10	0	0
pyraneu	1	1	0	25	0
senneu	1	0	0	0	25

从上表可知，测试的总体准确率为 93.85%。

对 7 类神经元进行分类，得到混淆矩阵如表 15 所示。

表 15 测试结果

样本 种类	biponeu	multineu	triponeu	motoneu	purkneu	pyraneu	senneu
biponeu	8	0	0	0	0	0	0
multineu	0	19	4	0	0	0	0
triponeu	0	9	4	0	0	0	1
motoneu	0	0	0	16	0	5	0
purkneu	0	0	0	10	0	0	0
pyraneu	0	0	0	1	0	26	0
senneu	0	0	1	0	0	0	25

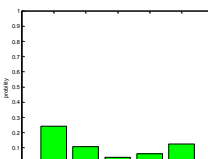
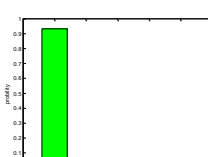
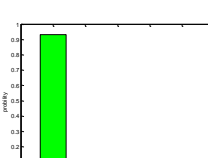
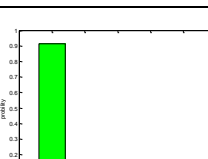
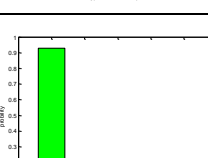
从上表可知，测试的总体准确率为 83.08%。

### 8.3 问题 2：类型识别及新类探测

出于与 7.4.2 节类似的考虑，我们同样通过考察待预测记录对于给定分类标签的输出概率来检测异常数据，从而获得可能存在的新的类别。

为了验证 SVM 模型对于该问题的有效性，我们利用测试集 T 进行检验。将 15 条记录输入神经网络分类器，得到的输出概率如表 16 所示。

表 16 模型测试结果

神经元种类	输出概率图	类型识别	是否正确
Calretinin		异类	正确
Calretinin		异类	正确
中间神经元 1		中间神经元	正确
中间神经元 2		中间神经元	正确
中间神经元 3		中间神经元	正确
中间神经元 4		中间神经元	正确
中间神经元 5		中间神经元	正确

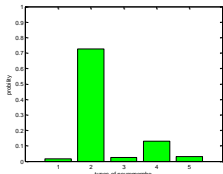
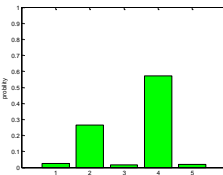
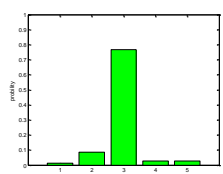
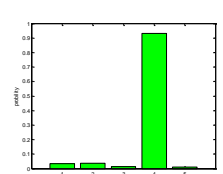
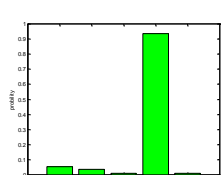
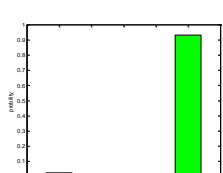
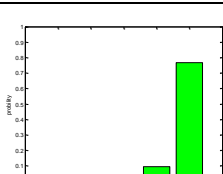
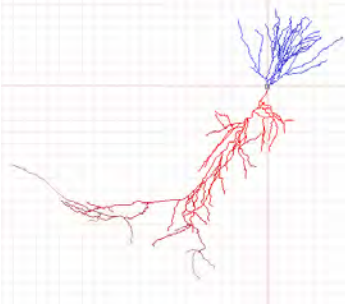
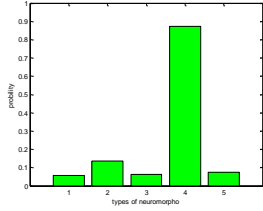
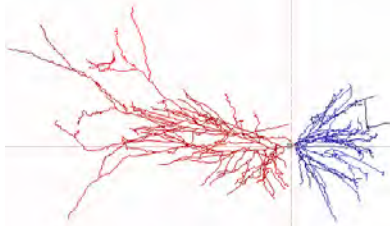
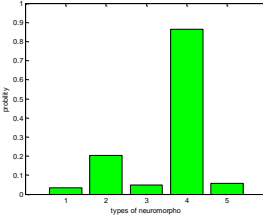
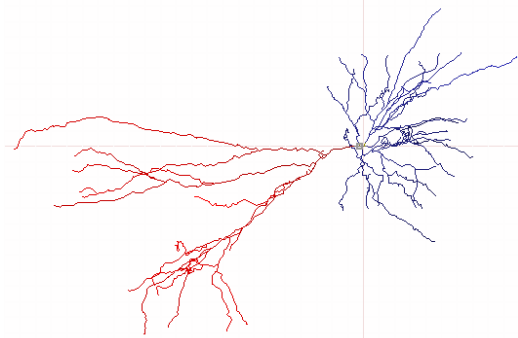
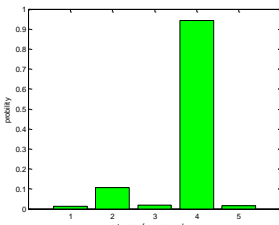
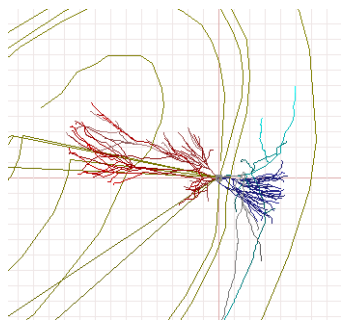
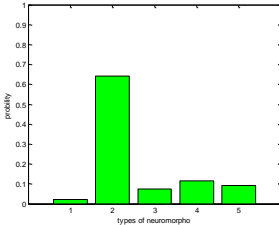
运动神经元 1		运动神经元	正确
运动神经元 2		锥体神经元	错误
普肯野神经元 1		普肯野神经元	正确
普肯野神经元 2		普肯野神经元	正确
锥体神经元 1		锥体神经元	正确
锥体神经元 2		锥体神经元	正确
感觉神经元 1		感觉神经元	正确
感觉神经元 2		感觉神经元	正确

表 16 所示的输出概率图，水平轴的条形图依次代表中间神经元、运动神经元、普肯野神经元、锥体神经元、感觉神经元。从表 16 中可以看出，测试集 T 输入到 SVM 分类器后，15 条记录共有 14 条识别正确，准确率约为 93.33%，与

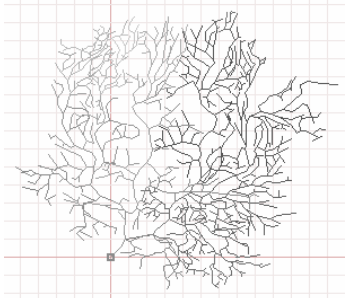
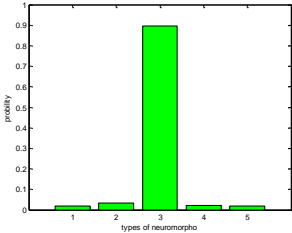
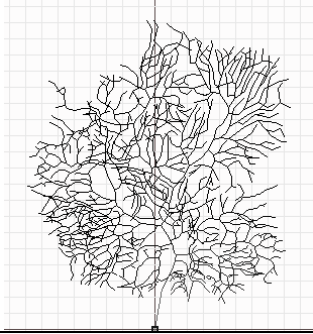
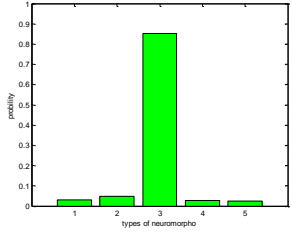
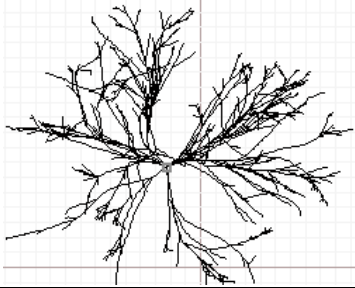
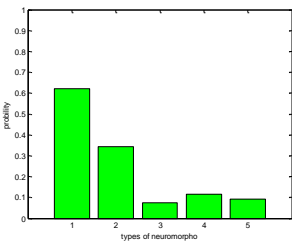
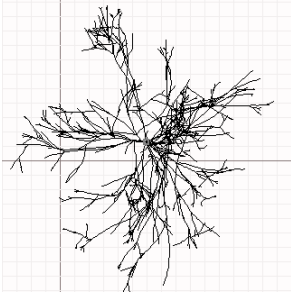
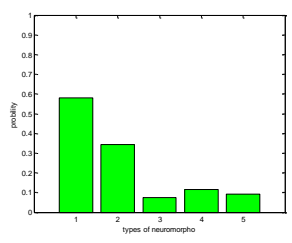
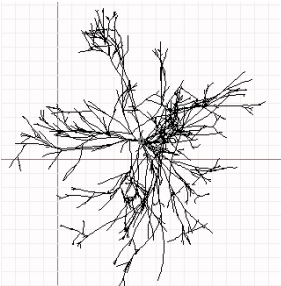
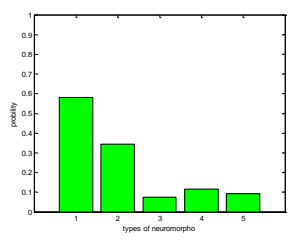
神经网络分类器相比较，正确率高。从而，可以得出结论，SVM 分类器对神经元的形态识别十分有效。

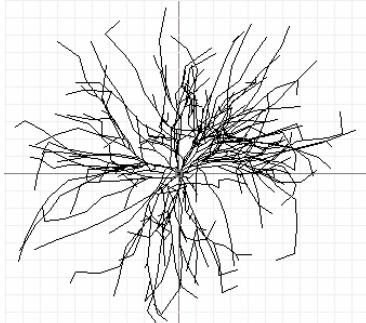
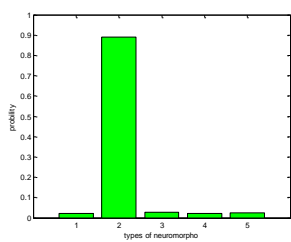
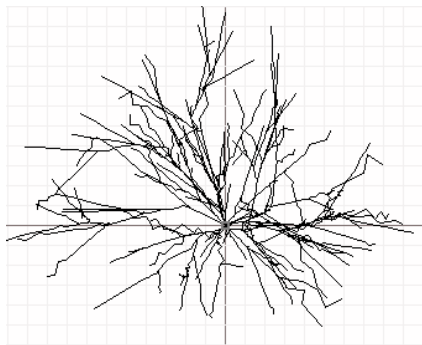
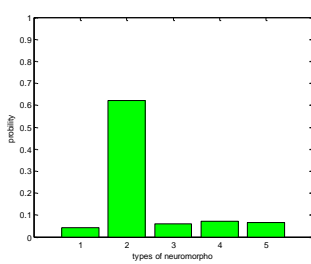
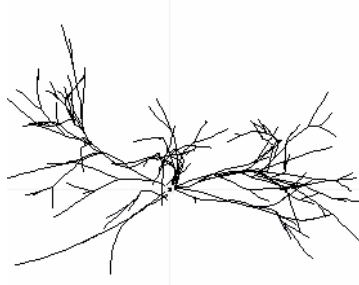
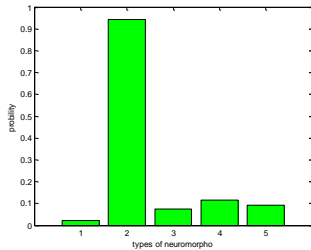
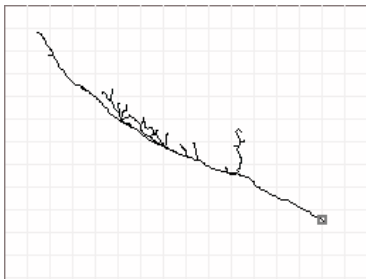
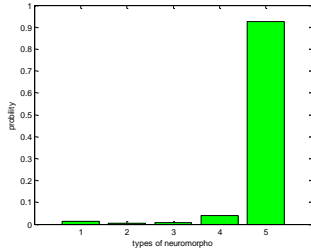
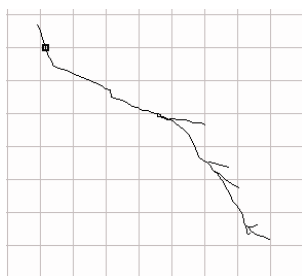
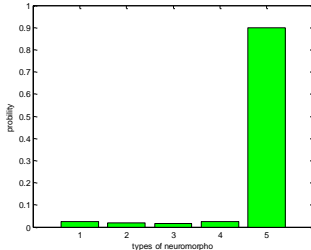
经过对 SVM 分类器的有效性检验，将测试集 B 中的 20 条记录输入到分类器中，进行类型识别及新类探测，得到输出概率如表 17 所示。

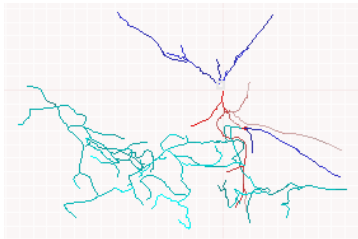
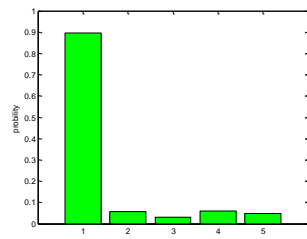
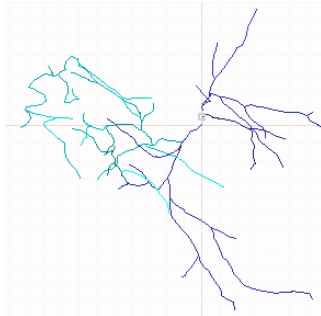
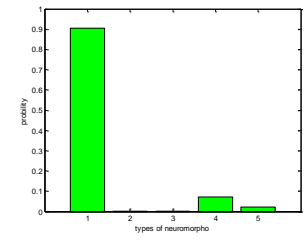

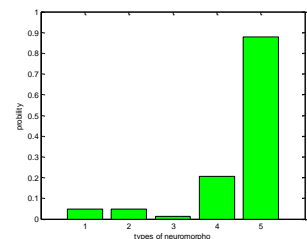
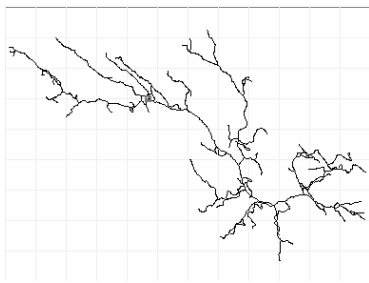
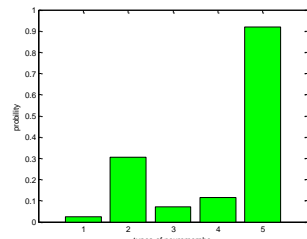
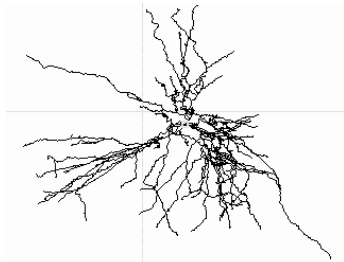
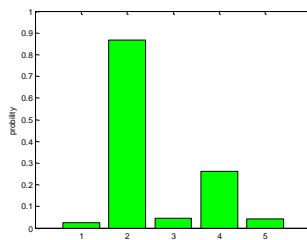

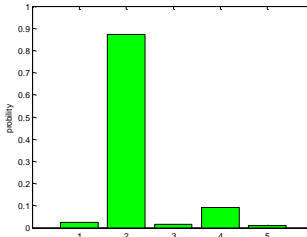
表 17 测试集 B 的类型识别及新类探测

神经元编号	神经元二维图	输出概率图	类型识别
1			椎体神经元
2			椎体神经元
3			椎体神经元
4			运动神经元



5			普肯野神经元
6			普肯野神经元
7			新类
8			新类
9			新类

10		 <table><caption>Probability distribution for neuron type 10</caption><thead><tr><th>types of neuromorpho</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.02</td></tr><tr><td>2</td><td>0.90</td></tr><tr><td>3</td><td>0.02</td></tr><tr><td>4</td><td>0.02</td></tr><tr><td>5</td><td>0.04</td></tr></tbody></table>	types of neuromorpho	probability	1	0.02	2	0.90	3	0.02	4	0.02	5	0.04	运动神经元
types of neuromorpho	probability														
1	0.02														
2	0.90														
3	0.02														
4	0.02														
5	0.04														
11		 <table><caption>Probability distribution for neuron type 11</caption><thead><tr><th>types of neuromorpho</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.05</td></tr><tr><td>2</td><td>0.65</td></tr><tr><td>3</td><td>0.05</td></tr><tr><td>4</td><td>0.08</td></tr><tr><td>5</td><td>0.07</td></tr></tbody></table>	types of neuromorpho	probability	1	0.05	2	0.65	3	0.05	4	0.08	5	0.07	运动神经元
types of neuromorpho	probability														
1	0.05														
2	0.65														
3	0.05														
4	0.08														
5	0.07														
12		 <table><caption>Probability distribution for neuron type 12</caption><thead><tr><th>types of neuromorpho</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.02</td></tr><tr><td>2</td><td>0.95</td></tr><tr><td>3</td><td>0.05</td></tr><tr><td>4</td><td>0.12</td></tr><tr><td>5</td><td>0.08</td></tr></tbody></table>	types of neuromorpho	probability	1	0.02	2	0.95	3	0.05	4	0.12	5	0.08	运动神经元
types of neuromorpho	probability														
1	0.02														
2	0.95														
3	0.05														
4	0.12														
5	0.08														
13		 <table><caption>Probability distribution for neuron type 13</caption><thead><tr><th>types of neuromorpho</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.02</td></tr><tr><td>2</td><td>0.01</td></tr><tr><td>3</td><td>0.01</td></tr><tr><td>4</td><td>0.05</td></tr><tr><td>5</td><td>0.95</td></tr></tbody></table>	types of neuromorpho	probability	1	0.02	2	0.01	3	0.01	4	0.05	5	0.95	感觉神经元
types of neuromorpho	probability														
1	0.02														
2	0.01														
3	0.01														
4	0.05														
5	0.95														
14		 <table><caption>Probability distribution for neuron type 14</caption><thead><tr><th>types of neuromorpho</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.02</td></tr><tr><td>2</td><td>0.01</td></tr><tr><td>3</td><td>0.01</td></tr><tr><td>4</td><td>0.02</td></tr><tr><td>5</td><td>0.92</td></tr></tbody></table>	types of neuromorpho	probability	1	0.02	2	0.01	3	0.01	4	0.02	5	0.92	感觉神经元
types of neuromorpho	probability														
1	0.02														
2	0.01														
3	0.01														
4	0.02														
5	0.92														

15		 <table border="1"><thead><tr><th>types of neuromorpha</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.9</td></tr><tr><td>2</td><td>0.05</td></tr><tr><td>3</td><td>0.02</td></tr><tr><td>4</td><td>0.05</td></tr><tr><td>5</td><td>0.03</td></tr></tbody></table>	types of neuromorpha	probability	1	0.9	2	0.05	3	0.02	4	0.05	5	0.03	中间神经元
types of neuromorpha	probability														
1	0.9														
2	0.05														
3	0.02														
4	0.05														
5	0.03														
16		 <table border="1"><thead><tr><th>types of neuromorpha</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.9</td></tr><tr><td>2</td><td>0.0</td></tr><tr><td>3</td><td>0.0</td></tr><tr><td>4</td><td>0.05</td></tr><tr><td>5</td><td>0.02</td></tr></tbody></table>	types of neuromorpha	probability	1	0.9	2	0.0	3	0.0	4	0.05	5	0.02	中间神经元
types of neuromorpha	probability														
1	0.9														
2	0.0														
3	0.0														
4	0.05														
5	0.02														
17		 <table border="1"><thead><tr><th>types of neuromorpha</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.05</td></tr><tr><td>2</td><td>0.05</td></tr><tr><td>3</td><td>0.02</td></tr><tr><td>4</td><td>0.2</td></tr><tr><td>5</td><td>0.9</td></tr></tbody></table>	types of neuromorpha	probability	1	0.05	2	0.05	3	0.02	4	0.2	5	0.9	感觉神经元
types of neuromorpha	probability														
1	0.05														
2	0.05														
3	0.02														
4	0.2														
5	0.9														
18		 <table border="1"><thead><tr><th>types of neuromorpha</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.02</td></tr><tr><td>2</td><td>0.3</td></tr><tr><td>3</td><td>0.05</td></tr><tr><td>4</td><td>0.1</td></tr><tr><td>5</td><td>0.9</td></tr></tbody></table>	types of neuromorpha	probability	1	0.02	2	0.3	3	0.05	4	0.1	5	0.9	感觉神经元
types of neuromorpha	probability														
1	0.02														
2	0.3														
3	0.05														
4	0.1														
5	0.9														
19		 <table border="1"><thead><tr><th>types of neuromorpha</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.02</td></tr><tr><td>2</td><td>0.85</td></tr><tr><td>3</td><td>0.05</td></tr><tr><td>4</td><td>0.25</td></tr><tr><td>5</td><td>0.05</td></tr></tbody></table>	types of neuromorpha	probability	1	0.02	2	0.85	3	0.05	4	0.25	5	0.05	运动神经元
types of neuromorpha	probability														
1	0.02														
2	0.85														
3	0.05														
4	0.25														
5	0.05														
20		 <table border="1"><thead><tr><th>types of neuromorpha</th><th>probability</th></tr></thead><tbody><tr><td>1</td><td>0.02</td></tr><tr><td>2</td><td>0.85</td></tr><tr><td>3</td><td>0.02</td></tr><tr><td>4</td><td>0.1</td></tr><tr><td>5</td><td>0.02</td></tr></tbody></table>	types of neuromorpha	probability	1	0.02	2	0.85	3	0.02	4	0.1	5	0.02	运动神经元
types of neuromorpha	probability														
1	0.02														
2	0.85														
3	0.02														
4	0.1														
5	0.02														

注：神经元二维图由 Neuromantic 软件得到。

根据表 17 可知，测试集 B 中共增加了 3 个新类，编号为 7，8，9。比较神经元 8，9 号的数据可知，8，9 号为同一条记录。通过观察编号 4 的二维图，可以看出，其二维图混乱，质疑其破碎程度比较高，有可能是由染色实验比较差所引起。因此，需通过对编号 4 的神经元进行具体分析，才能判断识别结果是否正确。

#### 8.4 问题三：命名建议

同神经网络模型类似，SVM 也无法给出显式的命名规则。因此我们考虑基于新类与已知类的相似度来命名原本不在我们分类体系中类别。具体方法如下：首先我们使用比较重要的 5 类神经元训练 SVM 分类器，使用 8.3 节描述的新类探测方法，当我们碰到新的不在现有分类体系中的类别时候时候，就提交给专家进行检查，根据 SVM 的输出概率我们可以确定它与分类体系中的哪一类最为相似，并根据这种相似性来对新类加以命名。例如表 16 中的 7、8、9 三类，我们就可以命名为“类运动中间神经元”，因为从它的概率输出来看，它和这两种的神经元有着比较大的相似性，这一点从 Neromantic 绘出的神经元二维图上也可以看出来。当然，这种方法可能对概率输出类似的不同的新类给出一样的命名，所以在命名过程中还需要专家的协助和干预。

#### 8.5 问题四：不同种类动物同种类别之间神经元区别

与神经网络模型类似，我们的 SVM 模型也具有这样的能力。采用 7.6 节提出的验证方式，我们在与其同样的数据集上进行 10 折交叉验证，得到混淆矩阵如表 18 所示。

表 18 测试结果

样本 种类	interneu	motoneu	pigpurkneu	ratpurkneu	pyraneu	senneu
interneu	22	0	0		0	0
motoneu	0	6	0		0	0
pigpurkneu	0	0	2	1	0	0
ratpurkneu	0	0	0	3	0	0
pyraneu	1	0	0	0	7	0
senneu	1	0	0	0	0	7

从上表可以看出，测试的总体准确率为 94%。

可以看到，考虑了同种神经元属于不同的动物种类后，SVM 仍然能够较为准确的将这些类别分开来，同时土拨鼠和鼠之间出现了类似神经网络模型中出现的分类错误，这也验证了我们在 7.6 节得出来的不同种动物中的同种中的神经元差异相对较小的结论。

## 九、模型四：神经元生长模型

### 9.1 L-System 建模

1968 年，美国的生物学家 Aristid Lindenmayer(1925 ~ 1989) 提出了 Lindenmayer 系统，简称 L-System。它是描述植物生长的数学模型，其基本思想可解释为理想化的树木生长过程：从一条树枝(种子)开始，发出新的芽枝，而发过芽枝的枝干又都发新芽枝……最后长出叶子。这一生长规律体现为斐波那契数列：1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …

记该序列的第  $n$  项为  $F_n$ ，则有递推关系式：

$$F_{n+2} = F_{n+1} + F_n \quad (n=1,2,3,\dots)$$

根据 L-System 的基本思想，我们先建立二维神经元的数学模型：从主干开始，然后沿着主干逐渐扩展到连接的枝干，再以递归的方式进行同样的过程，该过程持续到最终分枝。为简明起见，在这一模型中，每一枝干都化为一条直线，每次循环，枝干都被缩放，旋转和平移到其父枝的新位置上。

### 9.2 字符串替换

L-System 的核心思想是“字符串替换”。字符串即按一定规律的排布的字符集合。它可以包含短语、字母、数字或标点符号，一般用大写字母书写。字符串替换可以定义为根据一组改写规则或产生式(production)依次替换一个简单初始物的每一部分，即给出初始物（一条字符串）然后根据产生式规定的替换规则去替换初始字符串中的每一个字符。这种替换的次数是无穷的，可得到无限推导序列。替换中每一次反复称之为字符串的深度，例如深度为 3 就表明字符串替换进行了 3 次。以下是一个例子：

初始符  $\omega$ : B

产生式  $P_1: A \rightarrow AB$ ;  $P_2: B \rightarrow A$

深度	生成物
0	B
1	A
2	AB
3	ABA
4	ABAAB
5	ABAABABA
N	...

### 9.3 龟形建模

从字符串替换的概念可以看出，L-System 中的初始物和产生式均是由字符串来描述的。归根结底是因为 L-System 是一种形式语言。若要将 L-System 与神经元模拟联系起来，使之能表现真实神经元枝条的构造，就需给 L-System 中每一个字母赋予一个特定的几何图形含义。为了形象地说明，可以引进龟形这个概念。

最初的龟形来自于一个称为 LOGO 的内置了制图字符指令集的绘图软件。龟形的本质也就是一个运动指针。

龟形的主要思想是：将龟形状态定义成一个三元素集合  $(x, y, \alpha)$ ，其中坐标  $(x, y)$  表示龟形的位置，方向角  $\alpha$  表示龟形的方向，给出步长  $d$  和角增量  $\delta$ ，龟形行为对应于下列命令：

命令	作用	结果
$F(d)$	向前移动一步，步长为 $d$	龟形状态变为 $(x', y', \alpha)$ ，其中 $x' = x + d \cos \alpha$ $y' = y + d \sin \alpha$ ，在点 $(x, y)$ 和 $(x', y')$ 间画一直线段
$+(\delta)$	逆时针转 $\delta$	龟形的下一状态为 $(x, y, \alpha + \delta)$
$-(\delta)$	顺时针转 $\delta$	龟形的下一状态为 $(x, y, \alpha - \delta)$
$[$	将龟形的当前状态压入堆栈	将信息（包括龟形的位置和方向，可能还有其它属性，如所画线段的颜色及宽度）存入堆栈
$]$	从堆栈中弹出一个状态作为当前状态	不画线，但龟形的状态通常是改变的

给予龟形一条命令：FF[+F][F]，龟形即做以下操作：首先向正方向移动  $|2d|$  个单位，到达 A 点，然后把当前位置存入栈中，再逆时针旋转  $|\delta|$ ，并移动  $|d|$  个单位，出栈（回到 A 点）再入栈，最后顺时针旋转  $|\delta|$ ，并移动  $|d|$  个单位，出栈再回到 A 点。

上述为二维图例，若要绘制三维结构，需要将二维龟形命令扩展到三维空间上。首先给出描述龟形当前方向的三个向量  $\vec{H}, \vec{L}, \vec{U}$ ，它们分别表示龟形的向前、向左和向上的方向，均是单位长度，且满足方程  $\vec{H}' = R \vec{H}, \vec{L}' = R \vec{L}, \vec{U}' = R \vec{U}$ ，其中  $R$  是  $3 \times 3$  旋转矩阵。关于向量旋转  $\alpha$  的矩阵表示如下：

$$R_U(\alpha) = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_L(\alpha) = \begin{bmatrix} \cos \alpha & 0 & -\sin \alpha \\ 0 & 1 & 0 \\ \sin \alpha & 0 & \cos \alpha \end{bmatrix}$$

$$R_H(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

用下列带参数的符号控制龟形的空间方向：

$+(\delta)$ ：用旋转矩阵  $RU(\delta)$  表示向左转  $\delta$  角；

$-(\delta)$ ：用旋转矩阵  $RU(-\delta)$  表示向右转  $\delta$  角；

$\&(\delta)$ ：用旋转矩阵  $RL(\delta)$  表示向下转  $\delta$  角；

$\Lambda(\delta)$ ：用旋转矩阵  $RL(-\delta)$  表示向上转  $\delta$  角；

$\backslash(\delta)$ ：用旋转矩阵  $RH(\delta)$  表示左滚动  $\delta$  角；

$/(\delta)$ ：用旋转矩阵  $RH(-\delta)$  表示右滚动  $\delta$  角。

利用龟形建模的思想，先进行树的生长的模拟。

初始符：  $f$  ；

产生式：  $f \rightarrow ff -[-f + f + f] + [+f - f - f]$

分别迭代 2、3、4、5 后的图形如图 7 所示。

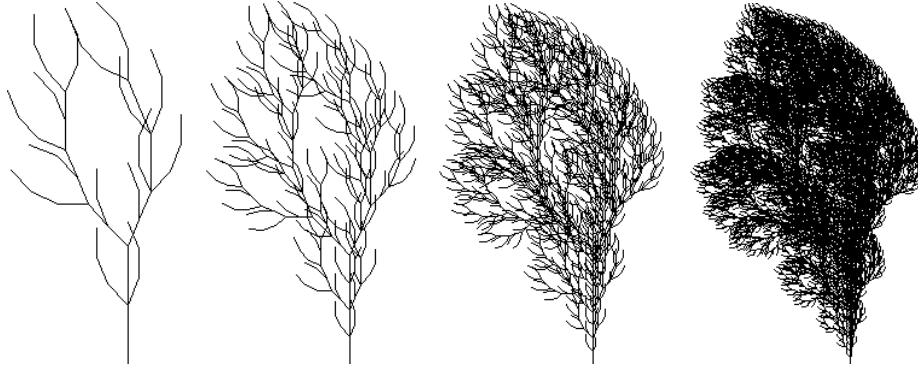


图 7 树生长模拟图

根据上图所示，利用 L-system 建模的思想可以很好的模拟出树形的生长过程。

#### 9.4 基于 L-System 的神经元生长的实现

根据上节的讨论可知，在假设 4 的条件下，如果知道神经元树突和轴突的生长变化规律，则可以通过 L-system 来模拟出神经元的生长过程。

图 8 所示的是模拟老鼠普肯野神经元在生长初期的变化[10]。



图 8 普肯野神经元生长过程

图 9 所示为仿真神经元的生长过程。



图 9 仿真神经元的生长变化

利用 L-system 预测出来的神经元生长三维图，提取神经元的特征参数，即可对其整个生长过程中的类型进行识别。

## 十、模型的评价

1、单一决策树模型简单，计算代价较小，能够快速的建立分类模型，同时对各类特征的提取与分类模型是在同一个过程中完成，抽取出来的特征也很简单，并且具有明确的物理意义，是人可以接受并且理解的，即使有些规则并不合理，它所提炼出来的规则也可以供人们制定分类标准时参考借鉴，同时决策树所提炼出来的指标还能给神经元的命名提供参考。

2、利用多棵决策树参照已有标准对神经元进行分类，减少了每步分步问题的目标类数，降低了分类问题的复杂性，在每一步上获得了相对较高的准确率。但是考虑到分类错误的后向传播，第一步分类的错误不仅会影响到最终结果，而且还有可能导致错误的积累放大，多决策树在总体分类效果上并不超越单一决策树的结果。

3、使用神经网络对神经元根据空间形态特征进行分类，在不同的数据集上



都能够获得相当高的分类准确性，随着数据集规模的增大，分类的性能可以得到提高。因此，可以通过增大数据集规模来提高分类器的性能。但是，此分类方法也有一定的缺陷，它没有明确的分类标准，而是把分类标准隐含在了网络结构当中，因此从网络外部很难直观的显示出输入数据的关键特征。

4、SVM 方法的优势在于物理意义相对明确，并且具有相对较好的泛化能力。对于本到题目而言，它具有和神经网络类似的优缺点，并且仅在原始数据集和扩展数据集上的小规模测试表明 SVM 的性能略低于神经网络模型。我们怀疑这是由于数据集过小且不平衡导致的，同时神经网络模型中可能存在了一定程度的过度拟合现象。这需要在更大规模的测试集合上验证工作。

总的来说，我们使用决策树模型解决了问题 1 和 3，同时指出它也具备一定解决问题 2 的能力；使用神经网络模型和 SVM 解决了问题 1、2、3、4，并得到了接近的实验结果，这两种模型的分类型对于本题的分类型性能也比较接近；但这两类对于显式样给出命名规则上都有一定的缺陷。由此可以看出，没有单一的模型能够系统完善的解决神经元根据空间形态分类问题，对于寻找完善的解决这一问题的方案，我们还需要做更多的工作。

## 参考文献

- [1] 刘深泉，姚良瑾等，神经元的形态识别和电位发放特性，第十二届全国非线性振动暨第九届全国非线性动力学和运动稳定性学术会议论文集, 2009.
- [2] G.A.Ascoli, R.F.G., Coordinatesystems for dendritic spines: a somatocentric approach, Complexity. 2(4): p. 40-48, 1997.
- [3] Mihail Bota, L.W.S., The neuron classification problem, Brain Reseach Reviews: p. 79-88, 2007.
- [4] 范明，孟小峰，数据挖掘，北京：机械工业出版社 2003.
- [5] Sanger, T., Probability density estimation for the interpretation of neural population codes, Journal of Neurophysiology, 1996.
- [6] Ascoli, G.A., L-neuron:A modeling tool fro the efficient genetation and parsimonious description of dendritic morphology, Neurocomputing. 32(33): p. 1003-1011, 2000.
- [7] J. Suykens, J.V., The support vector method of function estimation, Nonlinear Modeling: Advanced Black-Box Techniques: p. 55-86, 1998.
- [8] Wu, T.F.L., C.-J. Weng, Probability estimates for multi-class classification by pairwise coupling, JOURNAL OF MACHINE LEARNING RESEARCH. 5(2): p. 975-1006, 2005.
- [9] R.E. Fan, P.H.C., and C.J. Lin, Working set selection using the second order information for training SVM, Journal of Machine Learning Research. 6: p. 1889-1918, 2005.
- [10] P.Hamilton, A language to describe the growth of neurites, Biological Cybernetics. 68: p. 559-565, 1993.

## 附录

### 源程序

#### 决策树

```
package classifiers;
import java.util.Random;
import weka.classifiers.Evaluation;
import weka.classifiers.trees.*;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;
public class DecisionTree {
    String[] options;
    DataSource source;
    Instances data;
    J48 tree;
    public DecisionTree() {
        try {
            options = weka.core.Utils.splitOptions("-C 0.1 -M 2");
        } catch (Exception e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
        try {
            DataSource source = new
DataSource("/home/jiuren/mcm/experiment/dtree/130_7.arff");
            data = source.getDataSet();
            data.setClassIndex(0);
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        tree = new J48();
    }
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        DecisionTree dt = new DecisionTree();
        try {
            dt.tree.setOptions(dt.options);
            dt.tree.buildClassifier(dt.data);
            Evaluation eval = new Evaluation(dt.data);
```

```

        eval.crossValidateModel(dt.tree, dt.data, 10, new Random(1));
        System.out.println(eval.toSummaryString());
        //dt.tree.setAllowUnclassifiedInstances(true);

        Instances test = new
DataSource("/home/jiuren/mcm/experiment/dtree/Btest.arff").getDataSet();
        test.setClassIndex(0);

        for(int indexInstanceTest = 0; indexInstanceTest < test.numInstances();
++indexInstanceTest){
            System.out.println("evaluating "+ test.instance(indexInstanceTest) +
": ");
            for(double p :
dt.tree.distributionForInstance(test.instance(indexInstanceTest))){
                System.out.println(p);
            }
        }

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

## BP 神经网络

```

package classifiers;
import java.util.Random;
import weka.classifiers.Evaluation;
import weka.classifiers.functions.MultilayerPerceptron;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;
public class BPnetwork {
    /**
     * @param args
     */
    String[] options;
    DataSource source;
    Instances data;
    MultilayerPerceptron mp;
    public BPnetwork() {
        try {

```

```

        options = weka.core.Utils
            .splitOptions("-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a");
    } catch (Exception e1) {
        // TODO Auto-generated catch block
        e1.printStackTrace();
    }
    try {
        DataSource source = new
DataSource("/home/jiuren/mcm/experiment/bpnet/130_5.arff");
        data = source.getDataSet();
        data.setClassIndex(0);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    mp = new MultilayerPerceptron();
}
/**
 * @param args
 */
public static void main(String[] args) {
    // TODO Auto-generated method stub
    BPnetwork bpn = new BPnetwork();
    try {
        bpn.mp.setOptions(bpn.options);
        // bpn.mp.
        Evaluation eval = new Evaluation(bpn.data);
        eval.crossValidateModel(bpn.mp, bpn.data, 10, new Random(1));
        System.out.println(eval.toSummaryString());
        bpn.mp.buildClassifier(bpn.data);
        Instances test = new DataSource(
            "/home/jiuren/mcm/data/ppppp.arff").getDataSet();
        test.setClassIndex(0);
        for (int indexInstanceTest = 0; indexInstanceTest < test
            .numInstances(); ++indexInstanceTest) {
            System.out.println("evaluating "
                + test.instance(indexInstanceTest) + ": ");
            for (double p : bpn.mp.distributionForInstance(test
                .instance(indexInstanceTest))) {
                System.out.println(p);
            }
        }
    } catch (Exception e) {
        // TODO Auto-generated catch block

```

```

        e.printStackTrace();
    }
}
}

```

## SVM 程序

```

package classifiers;
import java.util.Random;
import weka.core.Instances;
import weka.classifiers.Evaluation;
import weka.classifiers.functions.LibSVM;
import weka.classifiers.functions.SMO;
import weka.classifiers.trees.J48;
import weka.classifiers.trees.RandomTree;
import weka.core.converters.ConverterUtils.DataSource;
public class SVM {
    /**
     * @param args
     */
    String[] options;
    DataSource source;
    Instances data;
    LibSVM svm;

    public SVM(){
        try {
            options = weka.core.Utils.splitOptions("-S 0 -K 2 -D 3 -G
1.7782794100389228 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.0010 -P 0.1 -Z -model
/home/jiuren/mcm");
            //options = weka.core.Utils.splitOptions("-S 0 -K 0 -D 3 -G 0.0 -R 0.0 -N
0.5 -M 40.0 -C 1.0 -E 0.0010 -P 0.1 -Z -B -model /home/jiuren/mcm");
            //options = weka.core.Utils.splitOptions("-S 0 -K 2 -D 3 -G
3.1622776601683795 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.0010 -P 0.1 -Z -model
/home/jiuren/mcm");
            //options = weka.core.Utils.splitOptions("-S 0 -K 2 -D 3 -G
3.1622776601683795 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.0010 -P 0.1 -Z -model
/home/jiuren/mcm");
        } catch (Exception e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }
}

```

```

        DataSource source = new DataSource("/home/jiuren/mcm/data/ppppp.arff");
        data = source.getDataSet();
        data.setClassIndex(0);
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    svm = new LibSVM();
}
/**
 * @param args
 */
public static void main(String[] args) {
    // TODO Auto-generated method stub

    SVM s= new SVM();
    try {
        s.svm.setOptions(s.options);
        s.svm.setProbabilityEstimates(true);
        Evaluation eval = new Evaluation(s.data);
        eval.crossValidateModel(s.svm, s.data,10, new Random(1));
        System.out.println(eval.toSummaryString());

        s.svm.buildClassifier(s.data);
        Instances test = new
DataSource("/home/jiuren/mcm/data/ppppp.arff").getDataSet();
        test.setClassIndex(0);
        for(int indexInstanceTest = 0; indexInstanceTest < test.numInstances();
++indexInstanceTest){
            System.out.println("evaluating "+ test.instance(indexInstanceTest) +
": ");
            for(double p :
s.svm.distributionForInstance(test.instance(indexInstanceTest))){
                System.out.println(p);
            }
        }

    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();}}}

```