摘要

植物的种类繁多，为研究植物分类方法，本文基于植物树叶的二值化图片数据，通过**解析几何计算**和**时间序列展开**的方法转换成轮廓特征向量和边缘特征向量；再通过**粒子群算法**优化的**深度神经网络**解决树叶识别分类问题，并分析核心指标对模型性能的影响，最终结合树叶纹理信息对模型进行改进和分析比较。

针对问题一，通过**解析几何计算**和**时间序列展开**，分别提取每一张图片中的特征向量。根据题目要求，本组首先利用 **matlab** 计算与图像形状相关的解析几何特征量，得到由八个几何学、拓扑学特征值组成的**八维**形状特征向量。然后将图像轮廓进行极化投影，并通过 **numpy 工具箱**将极化投影轮廓展开成时间序列，挖掘分析时间序列的三个特征值，组成**三维**边缘特征向量。最后合并两个向量得到**十一维**总体特征向量。

针对问题二，基于问题一中提取的特征信息，建立了 **PSO-DNN 网络**对树叶进行识别与分类。利用 **Keras 工具库**搭建**深度神经网络**，利用**粒子群算法**优化 DNN 网络的连接权值和阈值，对研究对象进行识别分类。训练后的神经网络模型预测准确率为 **91.037%**，并分析各指标的权重占比，判断出特征量中**时间序列熵、密实度和最大压痕深度**对模型性能影响最大，是模型的核心指标。将其剔除后，模型分类精确度损失率分别为 **6.937%**、**5.498%**、**3.559%**。

针对问题三，基于问题二中的 **PSO-DNN 网络**模型，将树叶纹理信息嵌入特征向量。增加网络输入层个数，并对模型进行参数调整，最终得到叶片的预测准确率达到 **96.634%**。

本文中所提到的模型优点主要有两点：一、提取的特征值信息包含量大、区分度高；二、利用 PSO 优化后的 DNN 网络全局收敛能力强，分类准确度高。

关键词： 时间序列展开　深度神经网络　粒子群算法　**Keras** 工具库

目录

一、 问题重述

1.1 问题背景

植物的种类繁多, 要了解和掌握如此多的植物, 必须进行一个科学的分类。人们常常根据植物的用途，或根据植物的一个或几个明显的形态进行分类，植物的识别与分类对于区分植物种类, 探索植物间的亲缘关系, 阐明植物系统的进化规律具有重要的意义。因此植物分类学是植物科学甚至整个生命科学的基础学科。目前对于树叶识别与分类主要由人完成, 但是树叶种类庞大, 依赖人工地进行树叶识别与分类是不现实的。所以树叶的研究对于植物总体的研究能提供很大的帮助。

从树叶的各个方面, 纹理, 硬度, 离心率等方面都可作为主要方向研究, 现对树叶的研究主要通过采集树叶图形, 利用数字图像处理来对树叶进行分类识别, 这种方法只停留在处理形态特征, 有很大的局限性, 忽略了树叶的生理特征和其他特征, 所以研究方法来综合处理树叶平面图像特征, 形态特征和生理特征很有必要性。

1.2 问题提出

围绕植物分类进行树叶识别与分类, 以树叶二值化图片为依据, 依次提出以下问题:

(1) 结合附件中的二值化的图片数据, 建立合适的图片数据提取方案, 量化处理图片数据, 并具体分析说明所提取数据信息的量化指标体系。
(2) 基于问题一中提取的数据信息, 建立合适的数学模型由数据出发判断叶子的种类, 研究判别分类的核心指标, 并估计出模型的性能以及核心指标对模型判别性能的影响。
(3) 基于二值化图片数据, 结合附件中叶子纹理的数据信息, 对原有模型进行改进, 并对新旧模型进行比较分析。

二、 模型假设

(1) 为了简化计算，假设题目中给出的模糊数据都是精确数据。
(2) 为了优化运算结果，假设所有全副武装的士兵都保持坐姿休息。
(3) 假设在战争中，装载消耗更少时间的优先度高于使用更少的民用船。
(4) 假设在装载过程中，同一泊口的船舰装载交替时间可以忽略不记。
(5) 假设港口被摧毁时，由于提前得到信息，港口上的船只与兵力没有损失，只是正在进行的装载工作停止，且进度完全损失。

三、 符号说明

符号	说明
I	研究图像
$A(I)$	图像面积
$C(I)$	图像几何中心
∂I	图像边界
$d(.)$	运算两点间欧式距离
ID	总特征向量
ID_{shape}	形状特征向量
ID_{margin}	边缘特征向量
id_k	研究图像特征值
r	极径
θ	极角
X	时间序列集
S	shapelet 子序列
x_i	神经网络第 i 个输入值
w_{ki}	第 i 个输入量连接的权值
b_k	神经网络阈值
f	激活函数
e	误差函数
E	全局误差
η	学习率
$c_{1,2}$	加速因子
β	惯性权重

四、 问题一模型的建立与求解

4.1 问题的描述与分析

问题一本质是一个多背包优化问题，要求使用尽可能少的背包，装载各种旅的一个旅编制兵力。本组根据题目要求，以派出船舰种类和数量作为决策向量，以所用的总船只数量为目标函数，以每艘船的面积装载限制作为约束条件。并设计沙石算法，使得每艘船面积利用率达到最高，从而使得调用船支数达到最小值。

4.2 模型的建立

4.2.1 预备工作

兵力、装备与船载的面积量化 计算附件 2 中的每种装备与所占面积，得到装备占用面积向量：

$$s_{xi} = l_i \times w_i \times \varepsilon_i \tag{1}$$

$$S_x = [s_{x1}, s_{x2}, \cdots, s_{x14}] \tag{2}$$

其中 s_{xi} 是装备 $X_i(i = 1, 2 \cdots, 14)$ 的占用面积，l_i、w_i 与 ε_i 分别是装备 X_i 的长、宽与面积修正系数。

计算附件 2 中每种旅一个营的占用面积，得到每营人口所占面积向量：

$$s_{pi} = p_i \times s \tag{3}$$

$$S_p = [s_{p1}, s_{p2}, \cdots, s_{p12}] \tag{4}$$

其中 p_i 是第 $i(i = 1, 2, \cdots, 12)$ 的全副武装人员数，取 $s = 0.5m^2$ 为每个全副武装人员占用的面积。

计算附件 3 中登陆舰可用装载面积，得到登陆舰有效面积向量：

$$s_{yi} = s_i \times \eta_1 \tag{5}$$

$$S_y = [s_{y1}, s_{y2}, \cdots, s_{y14}] \tag{6}$$

其中 s_{yi} 是登陆舰 $Y_i(i = 1, 2 \cdots, 14)$ 的可用装载面积，s_i 是登陆舰 Y_i 的总装载面积，取 $\eta_1 = 75\%$ 为登陆舰的有效面积率。

计算附件 3 中民用船可用装载面积，得到登陆舰有效面积向量：

$$S_y = [s_{y1}, s_{y2}, \cdots, s_{y14}] \tag{7}$$

$$s_{zj} = s_j \times \eta_2 \tag{8}$$

其中 s_{zi} 民用船 $Z_j(i = 1, 2 \cdots, 5)$ 的可用装载面积，s_j 是民用船 Z_j 的总装载面积，取 $\eta_2 = 70\%$ 为登陆舰的有效面积率。

4.2.2 优化函数与约束条件

决策向量为：

$$D = [y_1, y_2, \cdots, y_{14}, z_1, z_2, \cdots, z_5] \tag{9}$$

其中 y_k 是派出的登陆舰数量，z_k 是派出民用船的数量。

根据题意尽可能使用少的船舰数，即**目标函数**为使用船舰总数：

$$minZ = \sum_{1}^{19} D[k] \tag{10}$$

舰载面积系数向量为：

$$S_D = [S_y, S_z] \tag{11}$$

根据题目要求总舰载有效面积不小于总兵力装备占用面积：

$$D \cdot S_D \geq \sum S_x + \sum S_p \tag{12}$$

且由登陆舰 Y1 的有效舰载面积大于装备 X9(直升机) 的部队占用的面积：

$$y_1 \cdot s_{y1} \geq \sum s_{x9} + \sum s_{p6} \tag{13}$$

其中 $\sum s_{p6}$ 是唯一装备的部队——VI 旅的人口总和。

即使用最少船舰装载每种旅的各一个旅级编制的模型为：

$$minZ = \sum_{1}^{19} D[k] \tag{14}$$

$$s.t. \begin{cases} D \cdot S_D \geq \sum S_x + \sum S_p \\ y_1 \cdot s_{y1} \geq \sum s_{x9} + \sum s_{p6} \\ S_D = [S_y, S_z] \\ s_{xi} = l_i \times w_i \times \varepsilon_i \\ S_x = [s_{x1}, s_{x2}, \cdots, s_{x14}] \\ s_{pi} = p_i \times s \\ S_p = [s_{p1}, s_{p2}, \cdots, s_{p12}] \\ s_{yi} = s_i \times \eta_1 \\ S_y = [s_{y1}, s_{y2}, \cdots, s_{y14}] \\ s_{zj} = s_j \times \eta_2 \\ S_z = [s_{z1}, s_{z2}, \cdots, s_{z5}] \end{cases} \tag{15}$$

4.2.3 沙石算法

为使得调用的船舰数量最少，即需使得每艘船的未利用面积值达到最小：

$$min\Delta S = \sum S_x + \sum S_p - D \cdot S_D$$

其中 ΔS 为损失面积，本组设计沙石算法优化部队装载方案，使其得面积损失达到最小，具体如下：

算法思想 根据经验，使用沙砾和石子填满某个刚性容器，较好的方法是先装入石子，后灌入沙砾，可以让剩余空间达到最小。基于此，本组设计沙石算法，将带待装载部队(其占用面积不尽相同) 视作沙砾与石子，将船舰视作刚性容器，即可求出面积利用率最高的部队装载方案。

算法描述

(1) 装备均分：将每个旅分解为连编制，即得到面积规划向量：

$$A^k = [a_1^k, a_2^k, \cdots, a_n^k]$$

其中 a_i^k 的初始值是第 k 旅中的全副武装人员所占面积，其中 $k = 1, 2, \cdots, K$(K 为旅队编制总数)，$n = n_k$(n_k 为 k 旅连编制总数)。根据题目中装备均匀分配的要求，重复检索向量 A^k 中的每一个元素，选择最小值 $min\{a_i\}$ (装备量最少的营) 使得该连添加装备 X_j, 即令：

$$min\{a_i\} = min\{a_i\} + s_{xj}$$

重复检索直到该旅装备数 $X = 0$ 时,结束装备均分,得到装备均分后的部队 $\left\{A^k\right\}(k = 1, 2, \cdots, n_k)$($n_k$ 为 k 旅连编制总数)。

(2) 部队装载：将装备均分后的部队 $\left\{A^k\right\}$ 中的元素混合后由大到小进行排序得到排序后的营编制总部对数列 $\{T_n\}$:

$$T_n \geqslant T_{n+1}$$
$$n \leq \sum_{1}^{K} n_k$$

将所有派出船舰装载面积进行排序得到船舰面积数列 $\{P_n\}$：

$$P_n \geqslant P_{n+1}$$
$$n \leq \sum D$$

依次检索总部对数列 $\{T_n\}$，并依次检索数列 $\{P_n\}$ 对应的船舰，将其装入剩余面积足够的船舰，即若 $P_n \geqslant T_n$ 令：

$$P_n = P_n - T_n$$

重复检索直到部队装载完成，即当部队检索次数 $i = \sum_1^K n_k$ 时，算法结束。

(3) 结果输出：输出总共使用的船舰数量，即决策向量：

$$D = [y_1, y_2, \cdots, y_{14}, z_1, z_2, \cdots, z_5]$$

4.3 模型的求解

算法的实现

算法流程图

五、 问题二模型的建立与求解

5.1 问题的描述与分析

问题二要求针对附件 2 中的具体运输任务，制定使得装载时间最短，使用海上运输工具最少的兵力装载方案。本组通过遗传算法，将问题一中的决策向量作为染色体，装载时间和船支数量作为适应度函数，建立装载方案优化模型。修改问题一中的沙石算法，作为判断每个个体是否可以存活的验证算法，并随机生成 w 个能通过验证算法的初始个体。将某一代个体经过交叉、变异后的基因带入时间轴仿真模型，计算出装载所用时间作为适应度函数，将生成个体按照适应度函数大小进行第一次排序，再将装载所用时间大小相同的函数进行第二次排序，取出排序中前 w 个个体进行下一次进化，进化 g 代个体后可求得装载时间最短，同时使用海上运输工具最少的兵力装载方案。

5.2 模型的建立

5.2.1 优化函数与约束条件

基于问题一模型**决策向量**为：

$$D = [y_1, y_2, \cdots, y_{14}, z_1, z_2, \cdots, z_5] \tag{16}$$

目标函数为：

$$minZ = \sum_1^{19} D[k] \tag{17}$$

增加**目标函数**：

$$min\left\{\underset{k}{max}\sum t_i\right\} \tag{18}$$

其中 $\sum t_i(i=1,2\cdots,71)$，表示在共计 71 个泊口中第 i 个泊口的装载时间总和。即得到问题二优化模型：

$$minZ=\sum_{1}^{19}D[k] \tag{19}$$

$$min\left\{\underset{k}{max}\sum t_i\right\} \tag{20}$$

$$s.t.\begin{cases} D\cdot S_D\geq\sum S_x+\sum S_p \\ y_1\cdot s_{y1}\geq\sum s_{x9}+\sum s_{p6} \\ S_D=[S_y,S_z] \\ s_{xi}=l_i\times w_i\times\varepsilon_i \\ S_x=[s_{x1},s_{x2},\cdots,s_{x14}] \\ s_{pi}=p_i\times s \\ S_p=[s_{p1},s_{p2},\cdots,s_{p12}] \\ s_{yi}=s_i\times\eta_1 \\ S_y=[s_{y1},s_{y2},\cdots,s_{y14}] \\ s_{zj}=s_j\times\eta_2 \\ S_z=[s_{z1},s_{z2},\cdots,s_{z5}] \end{cases} \tag{21}$$

其中 $min\left\{\underset{k}{max}\sum t_i\right\}$ 可利用**泊口仿真模型**计算，并通过**遗传算法**算得全局最优解。

5.2.2 泊口仿真模型

建立泊口仿真模型以计算每种方案的装载时间，首先输入决策向量：

$$D=[y_1,y_2,\cdots,y_{14},z_1,z_2,\cdots,z_5]$$

定义时间函数为 t_0，初始化时间函数令 $t_0=0$，并令 t_0 以 1 为步长逐渐增加。定义港口决策向量为：

$$T=[t_1,t_2,\cdots,t_{71}]$$

$t_k(k=1,2,3,\cdots,71)$ 代表 71 个港口当前任务的结束时间。当 $t_k=t_0$ 时，表示港口 k 处于空闲状态，此时立即更新任务结束时间 t_k，即使得：

$$t_k=t_k+t_d$$

$$D[d]=D[d]-1$$

其中 t_d 是决策向量 D 中第 d 各元素对应船舰的装载时间，$D[d]$ 为决策向量 D 中的第 d 个元素。当

$$\begin{cases} \sum_1^{19} D[d] = 0 \\ t_0 \geqslant max t_k \end{cases}$$

即所有船支装载完成时，结束运算，输出当前的时间函数 t_0，即：

$$min\left\{\underset{k}{max}\sum t_i\right\} = t_0$$

5.2.3 遗传算法

编码 如图所示，每个个体由下面三个方块构成，其中最后一个进行遗传操作：

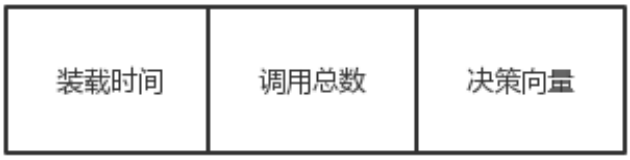

图 **1** 指标重要度占权图

根据题目要求，优先使用军用登陆舰，即决策向量 $D = [y_1, \cdots, y_{14}, z_1, \cdots, z_5]$ 中 $y_k(k = 1, 2, \cdots, 14)$ 为恒定值且等于其上限。故能简化决策向量为：

$$D = [z_1, z_2 \cdots, z_5]$$

交叉 为保证变异率并保留优秀基因片段和本题采用的两种交叉方式：

(1) ：单点交叉：对于两个父代个体 $D = [z_1, z_2 \cdots, z_5]$ 和 $D' = [z'_1, z'_2 \cdots, z'_5]$, 随机选择第 k 个基因处为交叉点，将该基因后所有基因进行交换，得到子代基因

(2) ：中间值交叉：对于两个父代个体 $D = [z_1, z_2 \cdots, z_5]$ 和 $D' = [z'_1, z'_2 \cdots, z'_5]$，随机选取 $z''_k \in [z_k, z'_k]$ 得到子代基因。

再选取交叉个体时采用混合分组的方法，将父代均匀混合后选取所有编号为奇数的个体，与其相邻对应编号为偶数的个体，通过两种交叉方式产生处两种类型的子代。

变异 为保证种群多样性，以 0.1 的变异率，对选择的个体执行变异操作。随机选择变异个体中的基因 z_k，使其值以各 50% 的概率加一或减一。

筛重 由于该模型基因维度较低，有大概率出现基因重复的个体，其对种群多样性有不利影响，并易使算法早熟。故在每次交叉变异生成新个体时删除重复基因，以提高算法的全局搜素能力。

选择 由于本题约束条件较多，且有两个包含两个适应度函数，故采用两种选择方式：

(1) ：约束淘汰：对于生成的个体基因 $D = [z_1, z_2 \cdots, z_5]$ 带入第一问的沙石算法中，计算部队是否可以完全装入船支中。若能则保留个体, 否则删除个体。

(2) ：精英选择：由于模型假设战场上使用较少的时间优先级高于使用更少船支，将每一代中的生成个体与上一代混合，的个体按照其**装载时间**由大到小进行**第一次排序**，再将装载时间相同的个体进行依照**调用总数**进行**第二次排序**。保留前列的 w 个个体，w 为设定的种群容纳量。

重复上述进化过程，当进化代数足够多时，即可得到全局最优解。

5.3 模型的求解

算法的实现

算法流程图

六、 问题三模型的建立与求解

6.1 问题的描述与分析

针对问题三，在问题二模型的基础上进行调整，当装载进行 24 小时后若干港口被摧毁，针对这一情况调整装载方案。

七、 模型的评价

7.1 模型的优点

(1) 利用时间序列提取树叶边缘信息，相较于传统特征量能更准确直观的表述边缘特征。

(2) PSO 优化后的 DNN 网络非线性映射能力强，泛化能力和稳定性明显高于一般神经网络，具有更强的鲁棒性。其全局收敛能力强，预测准确度高。

7.2 模型的缺点

树叶二值化图片信息提取不够完全，若能提取更高维度特征向量，分类准确度可进一步提高。

参考文献

[1] Tan Jing Wei,Chang Siow-Wee,Binti Abdul Kareem Sameem,Yap Hwa Jen,Yong Kien-Thai. Deep Learning for Plant Species Classification using Leaf Vein Morphometric.[J]. IEEE/ACM transactions on computational biology and bioinformatics,2018.

[2] Liu Jing,Sun Wanning,Su Yuting,Jing Peiguang,Yang Xiaokang. BE-CALF: Bit-Depth Enhancement by Concatenating All Level Features of DNN.[J]. IEEE transactions on image processing : a publication of the IEEE Signal Processing Society,2019,28(10).

[3] Matheus B. Vicari,Mathias Disney,Phil Wilkes,Andrew Burt,Kim Calders,William Woodgate. Leaf and wood classification framework for terrestrial LiDAR point clouds[J]. Methods in Ecology and Evolution,2019,10(5).

[4] 田德红, 何建敏. 基于变异粒子群优化与深度神经网络的航空弹药消耗预测模型 [J]. 南京理工大学学报,2018,42(06).

[5] 原继东, 王志海, 韩萌, 游洋. 基于逻辑 shapelets 转换的时间序列分类算法 [J]. 计算机学报,2015,38(07):1448-1459.

[6] 杨志辉, 胡红萍, 白艳萍. 基于主成分分析和 PSO-SVM 的树叶分类方法研究 [J]. 数学的实践与认识,2016,46(18):170-175.

[7] 侯铜, 姚立红, 阚江明. 基于叶片外形特征的植物识别研究 [J]. 湖南农业科学,2009(04):123-125+129.

[8] Febri Liantoni,Rifki Indra Perwira,Syahri Muharom,Riza Agung Firmansyah,Akhmad Fahruzi. Leaf classification with improved image feature based on the seven moment invariant[J]. Journal of Physics: Conference Series,2019,1175(1).

附录 A 代码

A.1 图片名称读取–matlab 源代码

```
p = genpath('.\data');% 获得文件夹data下所有子文件的路径，这些路径存在字符串p中，以';'分割
length_p = size(p,2);%字符串p的长度
path = {};%建立一个单元数组，数组的每个单元中包含一个目录
temp = [];
for i = 1:length_p %寻找分割符';'，一旦找到，则将路径temp写入path数组中
if p(i) ~= ';'
temp = [temp p(i)];
else
temp = [temp '\']; %在路径的最后加入 '\'
path = [path ; temp];
temp = [];
end
end
clear p length_p temp;
%至此获得data文件夹及其所有子文件夹（及子文件夹的子文件夹）的路径，存于数组path中。
%下面是逐一文件夹中读取图像
file_num = size(path,1);% 子文件夹的个数
for i = 1:file_num
file_path = path{i}; % 图像文件夹路径
img_path_list = dir(strcat(file_path,'*.jpg'));
img_num = length(img_path_list); %该文件夹中图像数量
if img_num > 0
for j = 1:img_num
image_name = img_path_list(j).name;% 图像名
%image = imread(strcat(file_path,image_name));
fprintf('%d %d %s\n',i-1,j,strcat(file_path,image_name));% 显示正在处理的路径和图像名
%图像处理过程 省略
%Untitled2(strcat(file_path,image_name));
end
end
end
```

A.2 树叶形状 id 计算–matlab 源代码

```
%UNTITLED2 Summary of this function goes here
%  Detailed explanation goes here
I=imread(str_path);
thresh = graythresh(I); %自动确定二值化阈值;
I1=im2bw(I);%二向化
stats = regionprops(I1,'Area','Eccentricity','Solidity','Perimeter', 'ConvexImage');
B = bwboundaries(I1,'noholes');
```

```
B=B{1,1};%边界坐标
A=stats.Area;%面积
P=stats.Perimeter;%周长
id(1)=stats.Eccentricity;%离心率
id(2)=stats.Solidity;%实密度
id(3)=4*pi*A/P.^2;%等周因子
temp1=zeros(size(B,1),size(B,1));
for i=1:size(B,1) %计算外接圆直径
for j=i:size(B,1)
temp1(i,j)=((B(i,1)-B(j,1)).^2+(B(i,2)-B(j,2)).^2).^0.5;
end
end
D=max(max(temp1)');%外接圆直径
temp2=zeros(size(B,1),1);
dd=zeros(size(I1,1),size(I1,2));
for i=1:size(I1,1)%计算内切圆半径
for j=1:size(I1,2)
if I1(i,j)==1
for k=1:size(B,1)
temp2(k,1)=((B(k,1)-i).^2+(B(k,2)-j).^2).^0.5;
end
dd(i,j)=min(temp2);
end
end
end
d=max(max(dd)');%内切圆半径
id(4)=1-2.*d./D;%伸长率
I3=stats.ConvexImage;
stats3 = regionprops(I3,'Centroid','Perimeter');
B3=bwboundaries(I3,'noholes');%凸型区域边界坐标
B3=B3{1,1};
C=stats3.Centroid;%凸型区域中心坐标
temp2=zeros(size(B3,1),1);
for k=1:size(B3,1)%计算最大距离
temp2(k,1)=((B3(k,1)-C(1,1)).^2+((B3(k,2)-C(1,2)).^2)).^0.5;
end
L1=max(temp2);%最大距离
temp2=zeros(size(B,1),1);
for k=1:size(B,1)%计算最大距离
temp2(k,1)=((B(k,1)-C(1,1)).^2+((B(k,2)-C(1,2)).^2)).^0.5;
end
L2=min(temp2);%最小距离
L=stats3.Perimeter;%凸型周长
id(5)=(L1-L2)./L;%最大压痕深度
for i=1:size(I1,1)
for j=1:size(I1,2)
if I1(i,j)==1
```

```
y1=i;
break;
end
end
end
for i=size(I1,1):-1:1
for j=size(I1,2):-1:1
if I1(i,j)==1
y2=i;
break;
end
end
end
for i=1:size(I1,2)
for j=1:size(I1,1)
if I1(j,i)==1
x1=i;
break;
end
end
end
for i=size(I1,2):-1:1
for j=size(I1,1):-1:1
if I1(j,i)==1
x2=i;
break;
end
end
end
id(6)=(x1-x2)./(y1-y2);%长宽比
fid=fopen('aaa.txt','a');%写入文件路径
fprintf(fid,'%f,%f,%f,%f,%f,%f\r\n',id);
fclose(fid);
end
```

A.3 时间序列展开–python 源代码

```
import pandas as pd

import scipy as sp
import scipy.ndimage as ndi
from scipy.signal import argrelextrema

import pandas as pd
```

```
from skimage import measure
from sklearn import metrics

import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches

from pylab import rcParams
rcParams['figure.figsize'] = (6, 6)


# ------------------------------------------------------ I/O ---

def read_img(img_no,str):
"""reads image from disk"""
return mpimg.imread(str)


def get_imgs(num, str):
"""convenience function, yields random sample from leaves"""
if type(num) == int:
imgs = range(1, 1600)
num = np.random.choice(imgs, size=num, replace=False)

for img_no in num:
yield img_no, preprocess(read_img(img_no,str))


# ------------------------------------------------ preprocessing ---

def threshold(img, threshold=250):
"""splits img to 0 and 255 values at threshold"""
return ((img > threshold) * 255).astype(img.dtype)


def portrait(img):
"""makes all leaves stand straight"""
y, x = np.shape(img)
return img.transpose() if x > y else img


def resample(img, size):
"""resamples img to size without distorsion"""
ratio = size / max(np.shape(img))
return sp.misc.imresize(img, ratio, mode='L', interp='nearest')
```

```
def fill(img, size=500, tolerance=0.95):
"""extends the image if it is signifficantly smaller than size"""
y, x = np.shape(img)

if x <= size * tolerance:
pad = np.zeros((y, int((size - x) / 2)), dtype=int)
img = np.concatenate((pad, img, pad), axis=1)

if y <= size * tolerance:
pad = np.zeros((int((size - y) / 2), x), dtype=int)
img = np.concatenate((pad, img, pad), axis=0)

return img

# ---------------------------------------------------- postprocessing ---

def standardize(arr1d):
"""move mean to zero, 1st SD to -1/+1"""
return (arr1d - arr1d.mean()) / arr1d.std()


def coords_to_cols(coords):
"""from x,y pairs to feature columns"""
return coords[::,1], coords[::,0]


def get_contour(img):
"""returns the coords of the longest contour"""
return max(measure.find_contours(img, .8), key=len)


def downsample_contour(coords, bins=512):
"""splits the array to ~equal bins, and returns one point per bin"""
edges = np.linspace(0, coords.shape[0],
num=bins).astype(int)
for b in range(bins-1):
yield [coords[edges[b]:edges[b+1],0].mean(),
coords[edges[b]:edges[b+1],1].mean()]


def get_center(img):
"""so that I do not have to remember the function ;)"""
return ndi.measurements.center_of_mass(img)

#------------------------------------------------------ feature engineering ---

def extract_shape(img):
```

```
"""
Expects prepared image, returns leaf shape in img format.
The strength of smoothing had to be dynamically set
in order to get consistent results for different sizes.
"""
size = int(np.count_nonzero(img)/1000)
brush = int(5 * size/size**.75)
return ndi.gaussian_filter(img, sigma=brush, mode='nearest') > 200


def near0_ix(timeseries_1d, radius=5):
"""finds near-zero values in time-series"""
return np.where(timeseries_1d < radius)[0]


def dist_line_line(src_arr, tgt_arr):
"""
returns 2 tgt_arr length arrays,
1st is distances, 2nd is src_arr indices
"""
return np.array(sp.spatial.cKDTree(src_arr).query(tgt_arr))


def dist_line_point(src_arr, point):
"""returns 1d array with distances from point"""
point1d = [[point[0], point[1]]] * len(src_arr)
return metrics.pairwise.paired_distances(src_arr, point1d)


def index_diff(kdt_output_1):
"""
Shows pairwise distance between all n and n+1 elements.
Useful to see, how the dist_line_line maps the two lines.
"""
return np.diff(kdt_output_1)


# ------------------------------------------------------ wrapping functions ---

# wrapper function for all preprocessing tasks
def preprocess(img, do_portrait=True, do_resample=500,
do_fill=True, do_threshold=250):
""" prepares image for processing"""
if do_portrait:
img = portrait(img)
if do_resample:
img = resample(img, size=do_resample)
```

```
if do_fill:
img = fill(img, size=do_resample)
if do_threshold:
img = threshold(img, threshold=do_threshold)

return img


# wrapper function for feature extraction tasks
def get_std_contours(img):
"""from image to standard-length countour pairs"""

# shape in boolean n:m format
blur = extract_shape(img)

# contours in [[x,y], ...] format
blade = np.array(list(downsample_contour(get_contour(img))))
shape = np.array(list(downsample_contour(get_contour(blur))))

# flagging blade points that fall inside the shape contour
# notice that we are loosing subpixel information here
blade_y, blade_x = coords_to_cols(blade)
blade_inv_ix = blur[blade_x.astype(int), blade_y.astype(int)]

# img and shape centers
shape_cy, shape_cx = get_center(blur)
blade_cy, blade_cx = get_center(img)

# img distance, shape distance (for time series plotting)
blade_dist = dist_line_line(shape, blade)
shape_dist = dist_line_point(shape, [shape_cx, shape_cy])

# fixing false + signs in the blade time series
blade_dist[0, blade_inv_ix] = blade_dist[0, blade_inv_ix] * -1

return {'shape_img': blur,
'shape_contour': shape,
'shape_center': (shape_cx, shape_cy),
'shape_series': [shape_dist, range(len(shape_dist))],
'blade_img': img,
'blade_contour': blade,
'blade_center': (blade_cx, blade_cy),
'blade_series': blade_dist,
'inversion_ix': blade_inv_ix}


# !/usr/bin/python
```

```
# -*- coding:utf-8 -*-


import os

outer_path = r'C:\Users\77526\PycharmProjects\untitled\Demo5\data'
folderlist = os.listdir(outer_path) # 列举文件夹


for folder in folderlist:
inner_path = os.path.join(outer_path, folder)
total_num_folder = len(folderlist) # 文件夹的总数
filelist = os.listdir(inner_path) # 列举图片
i = 0
for item in filelist:
total_num_file = len(filelist) # 单个文件夹内图片的总数
if item.endswith('.jpg'):
src = os.path.join(os.path.abspath(inner_path), item) # 原图的地址
# dst = os.path.join(os.path.abspath(inner_path), str(folder) + '_' + str(
#     i) + '.jpg') # 新图的地址（这里可以把str(folder) + '_' + str(i) + '.jpg'改成你想改的名称）
try:
# os.rename(src, dst)
print(src)
i += 1


title, img = list(get_imgs([968], src))[0]
features = get_std_contours(img)
print(features['blade_series'])
df = pd.DataFrame(features['blade_series'][0]).T
# df.to_csv('blade_series.csv', mode='a', header=False, index=False)
plt.plot(*coords_to_cols(features['shape_contour']))
plt.plot(*coords_to_cols(features['blade_contour']))
#plt.axis('equal')

plt.subplot(122)
plt.plot(*features['shape_series'])
plt.plot(*features['blade_series'])
plt.show()
except:
continue



# plt.plot(*coords_to_cols(features['shape_contour']))
# plt.plot(*coords_to_cols(features['blade_contour']))
# #plt.axis('equal')
```

```
#
# plt.subplot(122)
# plt.plot(*features['shape_series'])
# plt.plot(*features['blade_series'])
# plt.show()
```

A.4 边缘测试时间序列–C++ 源代码

```
#include "stdafx.h"
#include <opencv2/opencv.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <string>
#include <io.h>
#include <vector>
#include <iostream>
#include <math.h>

using namespace cv;
using namespace std;
char * filePath = "C:\\Users\\77526\\PycharmProjects\\untitled\\Demo5\\data";
//此处用的是斜杠，也可以用反斜
//但需注意的是由于C语言的特点，要用双反斜杠，即"E:\\MATLAB\\LBP\\scene_categories"
//cin >> folder; //也可以用此段代码直接在DOS窗口输入地址，此时只需正常的单反斜杠即可

using namespace std;

void getFiles(string foler, vector<string>& files);

int main() {
string folder = filePath;

vector<string> files;
getFiles(folder, files ); //files为返回的文件名构成的字符串向量组

for( int i = 0; i < files.size(); i++ ) { //files.size()返回文件数量
//To do here
get_arr(files[i]);
cout << files[i] << endl;
}
system("pause");
return 0;
}
```

```
void getFiles( string path, vector<string>& files ) {
//文件句柄
long hFile =  0;
//文件信息
struct _finddata_t fileinfo; //大家可以去查看一下_finddata结构组成
//以及_findfirst和_findnext的用法，了解后妈妈就再也不用担心我以后不会编了
string p;
if((hFile = _findfirst(p.assign(path).append("\\*").c_str(),&fileinfo)) != -1) {
do {
//如果是目录,迭代之
//如果不是,加入列表
if((fileinfo.attrib & _A_SUBDIR)) {
if(strcmp(fileinfo.name,".") != 0 && strcmp(fileinfo.name,"..") != 0)
getFiles( p.assign(path).append("\\").append(fileinfo.name), files );
}
else {
files.push_back(p.assign(path).append("\\").append(fileinfo.name) );
}
}
while(_findnext(hFile, &fileinfo) == 0);_findclose(hFile);
}
}



void get_arr(string str){
//读取图像
Mat src_image = imread(str);
//图像读取出错处理
if (!src_image.data)
{
cout << "src image load failed!" << endl;
return -1;
}
//显示源图像
namedWindow("原图", WINDOW_NORMAL);
imshow("原图", src_image);

//此处高斯去燥有助于后面二值化处理的效果
//Mat blur_image;
//GaussianBlur(src_image, blur_image, Size(15, 15), 0, 0);
//imshow("GaussianBlur", blur_image);

/*灰度变换与二值化*/
Mat gray_image, binary_image;
cvtColor(src_image, gray_image, COLOR_BGR2GRAY);
threshold(gray_image, binary_image, 30, 255, THRESH_BINARY | THRESH_TRIANGLE);
imshow("binary", binary_image);
```

```
/*形态学闭操作*/
Mat morph_image;
Mat kernel = getStructuringElement(MORPH_RECT, Size(3, 3), Point(-1, -1));
morphologyEx(binary_image, morph_image, MORPH_CLOSE, kernel, Point(-1, -1), 2);
imshow("morphology", morph_image);

/*查找外轮廓*/
vector< vector<Point> > contours;
vector<Vec4i> hireachy;
findContours(binary_image, contours, hireachy, CV_RETR_EXTERNAL, CHAIN_APPROX_NONE, Point());

int l;//目标轮廓索引
//寻找最大轮廓，即目标轮廓
for (size_t t = 0; t < contours.size(); t++)
{
/*过滤掉小的干扰轮廓*/
Rect rect = boundingRect(contours[t]);
if (rect.width < src_image.cols / 2)
continue;
//if (rect.width >(src_image.cols - 20))
l = t;//找到了目标轮廓，获取轮廓的索引
}
//画出目标轮廓
Mat result_image = Mat::zeros(src_image.size(), CV_8UC3);
vector< vector<Point> > draw_contours;
draw_contours.push_back(contours[l]);
drawContours(result_image, draw_contours, -1, Scalar(255,255,255), 1, 8, hireachy);
namedWindow("lunkuo", WINDOW_NORMAL);
imshow("lunkuo", result_image);

//计算轮廓的傅里叶描述子
Point p;
int x, y, s;
int i = 0,j = 0,u=0;
s = (int)contours[l].size();
Mat src1(Size(s,1),CV_8SC2);
float f[9000];//轮廓的实际描述子
float fd[16];//归一化后的描述子，并取前15个
for (u = 0; u < s; u++)
{
float sumx=0, sumy=0;
for (j = 0; j < s; j++)
{
p = contours[l].at(j);
x = p.x;
y = p.y;
```

```
sumx += (float)(x*cos(2*CV_PI*u*j/s) + y*sin(2 * CV_PI*u*j / s));
sumy+= (float)(y*cos(2 * CV_PI*u*j / s) - x*sin(2 * CV_PI*u*j / s));
}
src1.at<Vec2b>(0, u)[0] = sumx;
src1.at<Vec2b>(0, u)[1] = sumy;
f[u] = sqrt((sumx*sumx)+(sumy*sumy));
}
//傅立叶描述字的归一化
f[0] = 0;
fd[0] = 0;
for (int k = 2; k < 17; k++)
{
f[k] = f[k] / f[1];
fd[k - 1] = f[k];
cout << fd[k-1] << endl;
}
//保存数据
for (int k = 0; k < 16; k++)
{
FILE *fp = fopen("1.txt", "a");
fprintf(fp, "%8f,", fd[k]);
fclose(fp);
}
FILE *fp = fopen("1.txt", "a");
fprintf(fp, "\n");
fclose(fp);
waitKey();
}
```

A.5 数据可视化–python 源代码

```
from pyecharts import options as opts
from pyecharts.charts import Page, Pie


def pie_base(list) -> Pie:
c = (
Pie()
.add("", list, radius=["40%", "75%"],)
.set_global_opts(title_opts=opts.TitleOpts(title="各指标所占权重占比"),
legend_opts=opts.LegendOpts(
orient="vertical", pos_top="15%", pos_left="2%"

),          toolbox_opts=opts.ToolboxOpts(),
```

```
)
.set_series_opts(label_opts=opts.LabelOpts(formatter="{b}: {d}% "))

)
return c

def pie_rich_label(list) -> Pie:
c = (
Pie()
.add(
"",list,
radius=["40%", "75%"],
label_opts=opts.LabelOpts(
position="outside",
formatter=" {b|{b}: }{per|{d}%} ",
background_color="#eee",
border_color="#aaa",
border_width=1,
border_radius=4,
rich={
"a": {"color": "#999", "lineHeight": 22, "align": "center"},
"abg": {
"backgroundColor": "#e3e3e3",
"width": "100%",
"align": "right",
"height": 22,
"borderRadius": [4, 4, 0, 0],
},
"hr": {
"borderColor": "#aaa",
"width": "100%",
"borderWidth": 0.5,
"height": 0,
},
"b": {"fontSize": 16, "lineHeight": 33},
"per": {
"color": "#eee",
"backgroundColor": "#334455",
"padding": [2, 4],
"borderRadius": 2,
},
},
),
)
.set_global_opts(title_opts=opts.TitleOpts(title="各指标所占权重占比"),legend_opts=opts.LegendOpts(
orient="vertical", pos_top="15%", pos_left="2%"
```

```
),          toolbox_opts=opts.ToolboxOpts(),
)
)
return c

list = [['长宽比',0.294],
['离心率',0.227],
['密实度',0.328],
['等轴因子',0.276],
['伸长率',0.249],
['最大压痕深度',0.280],
['极小值点数',0.194],
['极大值点数',0.225],
['随机凸度',0.233],
['时间序列熵',0.429],
['最大压缩深度',0.214]]

c = pie_base(list)
c.render("xs.html")
d = pie_rich_label(list)
d.render("xss.html")
```

A.6 PSO-DNN 网络搭建–python 源代码

```
import pandas as pd
import numpy as np
import os

trainData = pd.read_csv(r'C:\Users\77526\Downloads\all_data.csv')
trainData = trainData.iloc[:,1:]
print(trainData.head())

from sklearn.utils import shuffle
trainData = shuffle(trainData)
trainData = trainData.values
y = trainData[:,0:1]
X= trainData[:,1:].astype(float)

y=pd.DataFrame(y, columns=['species'])
df = pd.get_dummies(y,columns=['species'])
species = [s.replace('species_', '') for s in df.columns.tolist()]
df.columns = species

y = df.values
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

from keras.models import Sequential #to initialize the neural network
from keras.layers import Dense # to build the layers of ANN
from keras.layers import Dropout


# Initialising the ANN
classifier = Sequential()

# Adding the input layer and the first hidden layer
classifier.add(Dense(units = 100, kernel_initializer ='uniform', activation = 'relu',
    input_dim = 192))
classifier.add(Dropout(0.2))

# Adding the second hidden layer
classifier.add(Dense(units = 100, kernel_initializer ='uniform', activation = 'relu'))
classifier.add(Dropout(0.2))

# Adding the third hidden layer
classifier.add(Dense(units = 100, kernel_initializer ='uniform', activation = 'relu'))

# Adding the output layer
classifier.add(Dense(units=99, kernel_initializer = 'uniform', activation = 'softmax'))

# Compiling the ANN
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
    ['accuracy'])

# Fitting the ANN to the Training set
model = classifier.fit(X_train, y_train, batch_size = 5, nb_epoch = 500)
print(model)


## Plotting the error with the number of iterations
## With each iteration the error reduces smoothly
import matplotlib.pyplot as plt
# define the function
def training_vis(hist):
loss = hist.history['loss']
val_loss = hist.history['val_loss']
```

```
acc = hist.history['acc']
val_acc = hist.history['val_acc']

# make a figure
fig = plt.figure(figsize=(8,4))
# subplot loss
ax1 = fig.add_subplot(121)
ax1.plot(loss,label='train_loss')
ax1.plot(val_loss,label='val_loss')
ax1.set_xlabel('Epochs')
ax1.set_ylabel('Loss')
ax1.set_title('Loss on Training and Validation Data')
ax1.legend()
# subplot acc
ax2 = fig.add_subplot(122)
ax2.plot(acc,label='train_acc')
ax2.plot(val_acc,label='val_acc')
ax2.set_xlabel('Epochs')
ax2.set_ylabel('Accuracy')
ax2.set_title('Accuracy on Training and Validation Data')
ax2.legend()
plt.tight_layout()
plt.show()


# call the function
training_vis(hist)
loss1 = hist.history['loss']
val_loss1 = hist.history['val_loss']
acc1 = hist.history['acc']
val_acc1 = hist.history['val_acc']



#计算权重
# loss,accuracy=classifier.evaluate(X_test,y_test)

# print('loss:',loss)
# print('accuracy:',accuracy)
weight_Dense_2,bias_Dense_3 = classifier.get_layer('input').get_weights()
weight_Dense_1,bias_Dense_3 = classifier.get_layer('output').get_weights()
df1 = pd.DataFrame(weight_Dense_1)
df2 = pd.DataFrame(weight_Dense_2)
df = df2.mul(df1,axis='columns',level=None, fill_value=None)
# df2.shape
x= abs(df2[1])
su =0
```

```
for i in range(0,15):
x= abs(df2.T[i])
su =su+sum(x)
print(sum(x)/100)
print(su)
```

A.7 各类分类器比较–python 源代码

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

def warn(*args, **kwargs): pass
import warnings
warnings.warn = warn

from sklearn.preprocessing import LabelEncoder
from sklearn.cross_validation import StratifiedShuffleSplit

train = pd.read_csv(r'C:\Users\77526\Downloads\all_data.csv')
# test = pd.read_csv('./test.csv')

def encode(train):
le = LabelEncoder().fit(train.species) #对数据进行标签编码
labels = le.transform(train.species)     # encode species strings
classes = list(le.classes_)              # save column names for submission
#     test_ids = test.id                       # save test ids for submission

train = train.drop(['species', 'id'], axis=1)
#     test = test.drop(['id'], axis=1)

return train, labels, classes

train, labels, classes = encode(train)
sss = StratifiedShuffleSplit(labels, 10, test_size=0.2, random_state=23)

for train_index, test_index in sss:
X_train, X_test = train.values[train_index], train.values[test_index]
y_train, y_test = labels[train_index], labels[test_index]

from sklearn.metrics import accuracy_score, log_loss
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,
```

```
    GradientBoostingClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis

classifiers = [
KNeighborsClassifier(3),
SVC(kernel="rbf", C=0.025, probability=True),
NuSVC(probability=True),
DecisionTreeClassifier(),
RandomForestClassifier(),
GradientBoostingClassifier(),
GaussianNB(),
LinearDiscriminantAnalysis(),]

<<<<<<< HEAD




=======
# Logging for Visual Comparison
log_cols=["Classifier", "Accuracy", "Log Loss"]
log = pd.DataFrame(columns=log_cols)

for clf in classifiers:
clf.fit(X_train, y_train)
name = clf.__class__.__name__

print("="*30)
print(name)

print('****Results****')
train_predictions = clf.predict(X_test)
acc = accuracy_score(y_test, train_predictions)
print("Accuracy: {:.4%}".format(acc))

train_predictions = clf.predict_proba(X_test)
ll = log_loss(y_test, train_predictions)
print("Log Loss: {}".format(ll))

log_entry = pd.DataFrame([[name, acc*100, ll]], columns=log_cols)
log = log.append(log_entry)

print("="*30)
sns.set_color_codes("muted")
sns.barplot(x='Accuracy', y='Classifier', data=log, color="b")
```

```
plt.xlabel('Accuracy %')
plt.title('Classifier Accuracy')
plt.show()

sns.set_color_codes("muted")
sns.barplot(x='Log Loss', y='Classifier', data=log, color="g")

plt.xlabel('Log Loss')
plt.title('Classifier Log Loss')
plt.show()
```