**实验名称：构造性覆盖算法的实现**

**实验要求：**实现构造性覆盖算法，并完成在规定数据集上的测试；

**可选项：**在完成覆盖算法的基础上，可以选择完成基于投票的覆盖算法，会适当加分。

**具体内容（版本 1-样本升维，且内积表示距离）：与原文一致的版本**

1. 数据集的预处理（主要可能包括：数据离散化，缺失值填充，归一化等）；
2. 数据升维（详见覆盖算法原始论文）；
3. 划分训练集和测试集；（Python、matlab 等均有现成的函数可用），当利用 C++、Java 等语言时可能需要自己编写数据集划分的代码，需要注意的是，当样本个数不能整除交叉数的时候，可以有一份数据多余或者少数其余的几份数据个数；
4. 编写训练过程的函数，需要注意的是：训练过程中用内积（inner product）代替传统的欧式距离函数，内积越大表示距离越小，反之，距离越大；最终的训练得到的模型用结构体表示，结构体内部应包括：每个覆盖的中心（样本）、覆盖的半径、覆盖的类别（标签），覆盖内样本个数，覆盖内具体的样本（这个可选，建议也包括）；**（核心步骤，逻辑不能乱，看提示）**
5. 编写测试过程的函数；测试过程可以采用多种策略，如：距离覆盖边界最近，距离覆盖中心最近以及万有引力的策略，上述策略可以参考附的中文文献，可以使用其中的任意一种，当然也可以给出所有策略下对应的算法指标。
6. 给出在指定数据集上算法的相应测试结果，最好以表格的形式给出，具体可以参考附录的相关文献的实验结果呈现方式。
7. （可选）完成基于投票的覆盖算法，具体参考附录的相应论文，给出相应结果。

**具体内容（版本 2-样本不升维，欧式距离度量样本远近）：**

1. 数据集的预处理（主要可能包括：数据离散化，缺失值填充，归一化等）；
2. 划分交叉集和验证集；（Python、matlab 等均有现成的函数可用），当利用 C++、Java 等语言时可能需要自己编写数据集划分的代码，需要注意的是，当样本个数不能整除交叉数的时候，可以有一份数据多余或者少数其余的几份数据个数；
3. 编写训练过程的函数，训练过程用欧式距离度量样本远近；最终的训练得到的模型用结构体表示，结构体内部应包括：每个覆盖的中心（样本）、覆盖的半径、覆盖的类别（标签），覆盖内样本个数，覆盖内具体的样本（这个可选，建议也包括）；**（核心步骤，逻辑不能乱，看提示）**
4. 编写测试过程的函数；测试过程可以采用多种策略，如：距离覆盖边界最近，距离覆盖中心最近以及万有引力策略，上述策略可以参考附的中文文献，可以使用其中的任意一种，当然也可以给出所有策略下对应的算法指标。
5. 给出在指定数据集上算法的相应测试结果，最好以表格的形式给出，具体可以参考附录的相关文献的实验结果呈现方式。
6. （可选）完成基于投票的覆盖算法，具体参考附录的相应论文，给出相应结果。

做两个版本中的任意一个都可以，都做也可，实验表明，两者性能无本质差距。

**提示：**算法实现过程最重要的是训练过程的逻辑，该算法的过程主要是一个寻找 d1（最近异类）和 d2（d1 约束下额最远同类，d2<d1）的过程，因为是一个学习算法，所以训练过程结束的标志是所有的样本都被学习了（即没有尚未被学习的样本了），此时学习过程，需要考虑 d1，d2 两个距离能不能找到的问题。若没有 d1 意味着什么，该如何处理，若没有 d2 又该如何处理。

**数据集：**从 UCI 数据集上下载分类的数据集即可，具体可以做 Voting based Constructive algorithm 一文中给出的数据集。

**该算法有几种变种：**

1）可以升维（版本 1），也可以不升维（版本 2）（注意：不升维的情况下，不可以使用内积作为距离度量，但是欧式距离仍然可以使用；升维的情况下欧式距离和内积都可以用，但是建议使用内积作为判断标准）；

2）学习覆盖的过程中，可以每学习一个覆盖就删除该覆盖的所有样本，也可以不删除已经学习过的样本，通过一个标志位来记录已经学习过的覆盖，这些被打过标志的样本，在下一次覆盖的过程中仍然用来约束覆盖的大小（这里的约束实际上是这些被覆盖的仍然参与求距离的计算）。但请注意，这些已经学习过的样本是不可能被后续的覆盖所包括的，因为我们采用的是折中半径的方法。学习过程样本是删除，还是用标志位记录，得到的覆盖是不同的，前者有大量多个覆盖都包括的重叠区域，后者不会有样本的重叠。前者复杂度更低，后者复杂度比较高。但两者在预测精度上差距并不明显。

**覆盖算法的网络结构**