

人工智能基础

编程作业 2

<http://staff.ustc.edu.cn/~linlixu/ai2019spring/ai2019spring.html>

提交截止时间：2019/7/3

助教：

盛鑫 [xins@mail.ustc.edu.cn]

赵若宇 [zry1997@mail.ustc.edu.cn]

辛媛 [lxjxy@mail.ustc.edu.cn]

滕思洁 [yunmo@mail.ustc.edu.cn]

实验说明

目的

本次实验基于机器学习中的监督学习和无监督学习来完成两个任务：国际象棋 checkmate 预测和青蛙聚类，请结合课上介绍的相关算法以及自己查阅拓展的一些算法，在给出的数据集上分别进行实验，以加强对相关算法原理及应用的理解。

提交

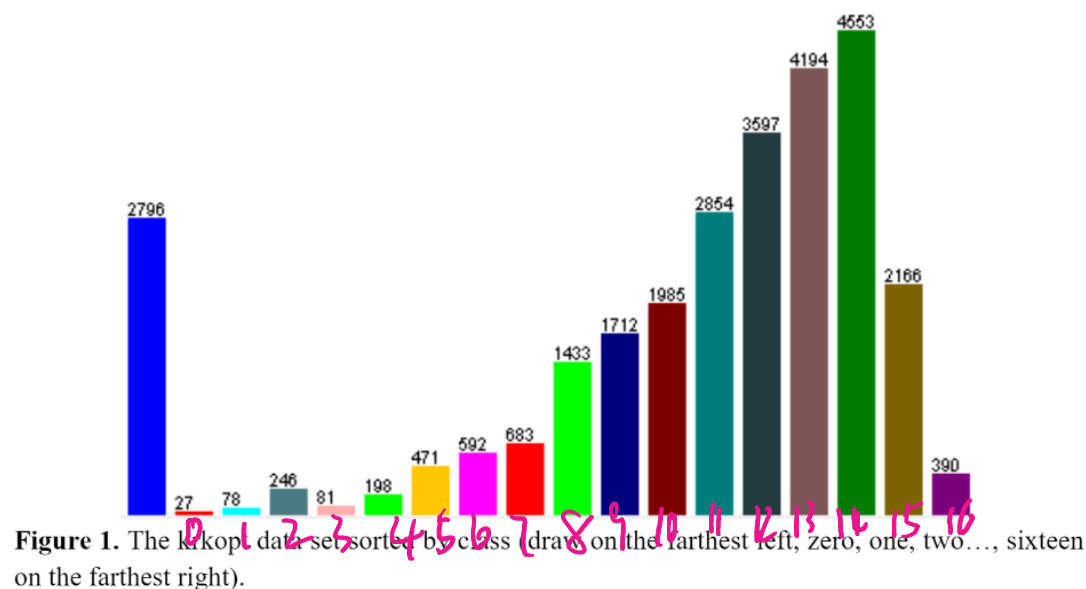
1. 提交邮箱：**ustc_ai2019@163.com**，主题：学号_姓名_实验二
2. 附件格式为“学号_姓名.(rar|zip)”，要包括实验报告和实验代码。两个实验的文件分别放在 part1 和 part2 文件夹中（两个文件夹中的所需提交内容会在实验要求中给出）。

Part 1. 国际象棋 Checkmate 预测 (60%)

数据集介绍： Krkopt 是一个国际象棋的残局数据集，在这张残局的棋盘上，只有白手国王 (White King)、白手车 (White Rook)、黑手国王 (Black King)。本次实验的任务是在给定前面所述的三者的位置的前提下，预测白手玩家能将军所需要的最少步数 (这里假设两个玩家的每步走法都是最优的)。

数据中包含 7 个属性 (含类别)，共有 28056 个样本，数据属性 (含类别) 如下所述：

1. 白手国王的列坐标 (White King Column)
2. 白手国王的行坐标 (White King Row)
3. 白手车的行坐标 (White Rook Column)
4. 白手车的列坐标 (White Rook Row)
5. 黑手国王的行坐标 (Black King Column)
6. 黑手国王的列坐标 (Black King Row)
7. **类别：** 最优步数 (optimal depth-of-win)，从 0~16 取值，若无法取胜，则为 draw，具体分布如下图所示



数据可于课程主页下载。由于考虑到数据集划分的随机性，所以我们已经将数据集划分成了训练集 trainset.csv 和测试集 testset.csv。

提示：

除类别外的 6 个属性不一定是直接作为单一属性来进行训练的，可以考虑属性之间的关联性，进行一定的处理，从而获得更好的实验效果。

训练与测试：

由于数据本身已经经过预处理，所以直接进行监督学习的训练即可。在监督学习中，训练数据集是带有标签（label）的，在本次实验中，即类别属性。

在训练过程中，需要从训练集 trainset 和对应的 trainlabel 中学习相应的多分类模型。

在测试过程中，用学习到的模型对测试集 testset 中的数据作预测，并将预测的结果和测试数据中的真实标签 testlabel 进行比较，从而度量学习的到的多分类模型的性能。

评价指标：

Accuracy（准确率），即正确预测的样本占所有测试样本的比重。

Macro F1：将 n 分类的评价拆成 n 个二分类的评价，计算每个二分类的 F1 score，n 个 F1 score 的平均值即为 Macro F1。

Micro F1：将 n 分类的评价拆成 n 个二分类的评价，将 n 个二分类评价的 TP、FP、RN 对应相加，计算评价准确率和召回率，由这 2 个准确率和召回率计算的 F1 score 即为 Micro F1。

Note：

$F1\ score = 2 * P * R / (P + R)$ ，其中准确率 $P = TP / (TP + FP)$ ，召回率 $R = TP / (TP + FN)$

真正例（True Positive，TP）：真实类别为正例，预测类别为正例。

假正例（False Positive，FP）：真实类别为负例，预测类别为正例。

假负例（False Negative，FN）：真实类别为正例，预测类别为负例。

真负例（True Negative，TN）：真实类别为负例，预测类别为负例。

预测值 实际值	Positive	Negative
正	TP	FN
负	FP	TN

实验要求：

1. 实现算法

①提交一个 KNN.py 文件，要求通过 **K 近邻算法**来解决多分类的问题：

实现一个 Python 函数 knn(trainset, trainlabel, testset, testlabel, k)，其中 k 为应选取的最近邻个数。要求函数返回对测试数据 testset 的预测 ypred，以及与 testlabel 进行比较后计算得到的性能指标 Accuracy、Macro F1 和 Micro F1。

②提交一个 decisionTree.py 文件，要求**调研决策树算法（ID3）并实现**来解决多分类的问题：

实现两个 Python 函数

chooseBestFeatrure(dataset)，利用信息熵实现选取特征，划分数数据集，计算得到当前最好的划分数数据集的特征，要求函数返回最好的特征 bestFeature。

createTree(trainset, trainlabel, testset, testlabel)，需调用 chooseBestFeature()函数。要求函数返回对测试数据 testset 的预测 ypred，以及与 testlabel 进行比较后计算得到的性能指标 Accuracy、Macro F1 和 Micro F1。

Note：需要给出最后训练出来的决策树**可视化结果**（可以使用 matplotlib 包实现）；

③提交一个 SVM.py 文件，要求通过**多分类 SVM 算法**来解决多分类的问题：

实现 Python 函数 multiClassSVM(trainset, trainlabel, testset, testlabel)，并要求实现嵌套函数 softSVM(trainset, trainlabel, sigma, C)，其中 C 为 soft margin SVM 的控制函数，

sigma 为控制核函数的参数，当 sigma=0 时，使用线性核函数 $K(x_i, x_j) = x_i^T x_j$ ，其他

情况则使用 RBF 核函数 $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{sigma^2}}$ 。要求函数返回对测试数据 testset 的预测 ypred，以及与 testlabel 进行比较后计算得到的性能指标 Accuracy、Macro F1 和 Micro F1。

提示：传统 SVM 仅能实现二分类问题，但是可以将多分类的问题看作二分类问题的一个扩展，训练时依次把某个类别的样本归为一类，其他剩余的样本归为另一类。这样 k 个类别的样本就构造出了 k 个 SVM。分类时将未知样本分类为具有最大分类函数值的那类（这里可以将 softSVM 函数封装，分别调用 k 次来训练得到 k 个 SVM 即可）。

Note：三个算法得分依次占比 **10%、15%、15%**

2. (10%) 在数据集上使用交叉验证法来进行训练集和验证集的划分及训练 (使用 5-fold 交叉验证), 同时为每个算法挑选合适的参数。

对每个算法:

- a. 返回一个矩阵, 表示每一个参数 (参数组合) 在每一个 fold 上的 Micro F1 值 (若有 10 个参数, 则返回 10x5 的矩阵)
- b. 挑选在 5 个 fold 中平均 Micro F1 值最高的参数 (参数组合)

在实验报告中需要记录交叉验证的结果, 即对于每个参数 (参数组合) 在 5 个 fold 上的平均 Micro F1 值。

注: 交叉验证过程中不能使用测试数据。对于 SVM, sigma 的选取范围可以以数据的平均距离 $\text{avgdist} = \frac{\sum_{i,j=1}^n \|x_i - x_j\|^2}{n^2}$ 为基础进行放缩, 例如在 $\text{avgdist} \times [\dots 10^{-2}, 10^{-1}, 1, 10, 10^2, \dots]$ 中选择, 放缩范围可以自己调整。

3. 语言要求: 使用 Python2.x 或 3.x (注意, 不可以直接调用机器学习的包, 如 sklearn)

提交报告要求 (10%):

1. 给出你对各个属性间关系进行处理的思路 (若是直接进行训练, 可以不写);
2. 分别给出算法的伪代码;
3. 根据评价指标, 给出模型评估结果, 要求给出对应的图表分析。

提交内容:

1. 所有的源码 (KNN.py、decisionTree.py、SVM.py), 源码保存在文件夹 src 中;
2. 实验报告保存为 report1.pdf, 置于根目录下。

Part 2.对青蛙进行聚类(40%)

在这部分实验中, 我们将完成根据青蛙声音对青蛙的聚类。输入是 7195 个 22 维的青蛙

声音特征向量，已经进行过归一化处理，我们需要使用以下几个聚类算法进行聚类。

数据集及数据处理说明：

青蛙的声音特征保存在 Frogs_MFCCs.csv 文件中，前 22 列是归一化处理后的 22 维声音特征向量，接着的三列是 Family、Genus 和 Species 三个标签，每个样例都有对应的三个标签，大家可手动删去其中任意两个，留下一个作为标准结果。RecordID 可直接忽略。

实验要求：

1. 实现一个 KMeans 聚类算法（10%）

实现并提交一个 Python 函数 KMeans (k, data)，算法要求自己实现。其中 data 是要进行聚类的数据，k 是类别数目。要求返回纯度 purity 以及兰德指数 RI，(purity 和 RI 以 tuple 形式返回)，并将聚类结果写入.csv 文件中。

2. 实现一个 PCA 降维算法（10%）

提交一个 Python 函数 PCA (data, threshold) 其中 threshold 表示特征值的累计贡献率。即选择前 m 个特征向量，使得

$$\frac{\text{Sum}(\text{first } m - 1 \text{ eigenvalues})}{\text{Sum}(\text{all eigenvalues})} < threshold \leq \frac{\text{Sum}(\text{first } m \text{ eigenvalues})}{\text{Sum}(\text{all eigenvalues})}$$

返回值为降维后的矩阵。

然后对降维后的数据再次调用 KMeans 聚类算法得到结果，并将聚类结果写入.csv 文件中。

同时需要将降维后的结果进行**可视化**（形成 2 维的散点图）。

3. 拓展内容：调研并实现层次聚类或 DBSCAN（10%），在两个算法中**任选一个**完成。

a. 层次聚类算法：

层次聚类算法分凝聚式和分裂式两种。其中**分裂式**采用自顶向下的思想，先将所有样本都看成一个簇，然后通过迭代将其分成更小的簇。**凝聚式**采用的是自底向上的思想，先将每一个样本都看成一个不同的簇，通过重复将最近的簇合并来进行聚类，直到类的数目达到指定要求。**本次实验要求采用凝聚式层次聚类**，其步骤如下：

- (1) 将每个对象看作一类，计算两两之间的最小距离；
- (2) 将距离最小的两个类合并成一个新类；
- (3) 重新计算新类与所有类之间的距离；
- (4) 重复(2)、(3)，直到所有类最后类数达到指定值。

b. **DBSCAN** 算法：

DBSCAN 是一种基于密度的聚类算法，这类密度聚类算法一般假定类别可以通过样本分布的紧密程度决定。同一类别的样本，它们之间的紧密相连的，也就是说，在该类别任意样本周围不远处一定有同类别的样本存在。通过将紧密相连的样本划为一类，这样就得到了一个聚类类别。通过将所有各组紧密相连的样本划为各个不同的类别，则我们就得到了最终的所有聚类类别结果。

DBSCAN 是基于一组邻域来描述样本集的紧密程度的，参数(ϵ , MinPts)用来描述邻域的样本分布紧密程度。其中， ϵ 描述了某一样本的邻域距离阈值，MinPts 描述了某一样本的距离为 ϵ 的邻域中样本个数的阈值。

假设样本集是 $D=(x_1, x_2, \dots, x_n)$, 则 DBSCAN 具体的密度描述定义如下：

- 1) ϵ -邻域：对于 $x_i \in D$ ，其 ϵ -邻域包含样本集 D 中与 x_i 的距离不大于 ϵ 的子样本集，即 $N_\epsilon(x_i) = \{x_j \in D | \text{distance}(x_i, x_j) \leq \epsilon\}$ ，这个子样本集的个数记为 $|N_\epsilon(x_i)|$ 。
- 2) 核心对象：对于任一样本 $x_i \in D$ ，如果其 ϵ -邻域对应的 $N_\epsilon(x_i)$ 至少包含 MinPts 个样本，即如果 $|N_\epsilon(x_i)| \geq \text{MinPts}$ ，则 x_i 是核心对象。
- 3) 密度直达：如果 x_i 位于 x_j 的 ϵ -邻域中，且 x_j 是核心对象，则称 x_i 由 x_j 密度直达。注意反之不一定成立，即此时不能说 x_j 由 x_i 密度直达，除非且 x_i 也是核心对象。
- 4) 密度可达：对于 x_i 和 x_j ，如果存在样本序列 p_1, p_2, \dots, p_T ，满足 $p_1 = x_i, p_T = x_j$ ，且 p_{t+1} 由 p_t 密度直达，则称 x_j 由 x_i 密度可达。也就是说，密度可达满足传递性。此时序列中的传递样本 p_1, p_2, \dots, p_{T-1} 均为核心对象，因为只有核心对象才能使其他样本密度直达。注意密度可达也不满足对称性，这个可以由密度直达的不对称性得出。
- 5) 密度相连：对于 x_i 和 x_j ，如果存在核心对象样本 x_k ，使 x_i 和 x_j 均由 x_k 密度可达，则称

x_i 和 x_j 密度相连。注意密度相连关系是满足对称性的。

算法流程：

输入：样本集 $D=(x_1, x_2, \dots, x_n)$ ，邻域参数 $(\epsilon, \text{MinPts})$ ，样本距离度量方式

输出：簇划分 K

(1) 初始化核心对象集合 $\Omega=\emptyset$ ，初始化聚类簇数 $w=0$ ，初始化未访问样本集合 $\Gamma = D$ ，簇划分 $K = \emptyset$

(2) 对于 $j=1, 2, \dots, n$ ，按下面的步骤找出所有的核心对象：

(2.1) 通过距离度量方式，找到样本 x_j 的 ϵ -邻域子样本集 $N_\epsilon(x_j)$

(2.2) 如果子样本集样本个数满足 $|N_\epsilon(x_j)| \geq \text{MinPts}$ ，将样本 x_j 加入核心对象样本集合：

$\Omega = \Omega \cup \{x_j\}$

(3) 如果核心对象集合 $\Omega=\emptyset$ ，则算法结束，否则转入步骤(4)

(4) 在核心对象集合 Ω 中，随机选择一个核心对象 o ，初始化当前簇核心对象队列 $\Omega_{\text{cur}}=\{o\}$ ，初始化类别序号 $w=w+1$ ，初始化当前簇样本集合 $K_w=\{o\}$ ，更新未访问样本集合 $\Gamma=\Gamma-\{o\}$

(5) 如果当前簇核心对象队列 $\Omega_{\text{cur}}=\emptyset$ ，则当前聚类簇 K_w 生成完毕，更新簇划分 $K=\{K_1, K_2, \dots, K_w\}$ ，更新核心对象集合 $\Omega=\Omega-K_w$ ，转入步骤(3)

(6) 在当前簇核心对象队列 Ω_{cur} 中取出一个核心对象 o' ，通过邻域距离阈值 ϵ 找出所有的 ϵ -邻域子样本集 $N_\epsilon(o')$ ，令 $\Delta=N_\epsilon(o') \cap \Gamma$ ，更新当前簇样本集合 $K_w=K_w \cup \Delta$ ，更新未访问样本集合 $\Gamma=\Gamma-\Delta$ ，更新 $\Omega_{\text{cur}}=\Omega_{\text{cur}} \cup (\Delta \cap \Omega)-o'$ ，转入步骤(5)

输出结果为：簇划分 $K=\{K_1, K_2, \dots, K_w\}$

如果使用层次聚类算法要实现函数 $\text{HC}(\text{data}, n_clusters)$ ，其中 $n_clusters$ 为簇的个数。

如果使用 DBSCAN 算法，需要实现函数 $\text{DBSCAN}(\text{data}, \text{eps}, \text{minPts})$ ，其中 eps 为邻域距离阈值， MinPts 描述了某一样本的距离为 eps 的邻域中样本个数的阈值。

无论选择哪种算法都要求返回纯度 purity 以及兰德指数 RI ，(purity 和 RI 以 tuple 形式返回)，并将聚类结果写在.csv 文件中。

4. 语言要求：使用 Python2.x 或 3.x（注意，**不可以直接调用机器学习的包**，如 sklearn）

3. 评价指标

纯度

$$\text{purity} = \frac{1}{N} \sum_w \max_j |K_w \cap c_j|$$

其中 N 代表元素总数， K_w 代表第 w 个聚类， $C = \{c_1, c_2, \dots, c_j\}$ 是真实的分组集合， c_j 表示第 j 个分类。

兰德系数

兰德系数为

$$RI = \frac{a + d}{a + b + c + d}$$

假设用 C 表示真实的分组情况， K 表示聚类结果，那么：

a 为在 C 中为同一类且在 K 中也为同一类别的数据点对数

b 为在 C 中为同一类但在 K 中却隶属于不同类别的数据点对数

c 为在 C 中不在同一类但在 K 中为同一类别的数据点对数

d 为在 C 中不在同一类且在 K 中也不属于同一类别的数据点对数

显然 $a+b+c+d$ 为总的的数据点对数，可用 C_n^2 计算，其中 n 为元素总数。

RI 取值范围为 $[0,1]$ ，值越大意味着聚类结果与真实情况越吻合。

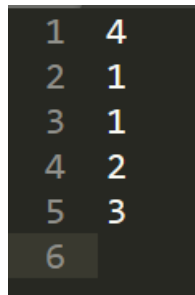
总结以上实验，并对结果进行分析。其中 KMeans 实验需要给出选择 k 的分析过程。

4. 实验报告（10%）：

1. 给出 KMeans 算法 k 选择的依据；
2. 在 KMeans 算法实验中，比较并分析 PCA 降维前后的纯度和兰德指数，给出图表展示和结果分析（包括降维后的**散点图**）；
3. 给出 DBSCAN 算法或层级聚类算法的图表分析。

5. 提交要求：

1. 需要提交算法代码, KMeans.py, KMeans_PCA.py 以及 HC.py 和 DBSCAN.py 二选一, 保存在 src 文件夹中;
2. 需要提交聚类结果, KMeans.csv, KMeans_PCA.csv 以及 HC.csv 和 DBSCAN.csv 二选一。结果中, 第一行标明类别数目, 剩下每行为 Frogs_MFCCs.csv 文件中对应行的聚类结果, 用数字 (1-K) 表示。比如 KMeans 算法 $K=4$ 时, 前 2 个青蛙同类, 而之后的两个分属另外两类, 那么输出文件 KMeans.csv 为:



1	4
2	1
3	1
4	2
5	3
6	

所有的 csv 文件保存在 result 文件夹中;

3. 需要提交报告保存为 report2.pdf, 保存在根目录下。