人工智能导论作业 3：Aliens 游戏

石睿 (211300024、211300024@smail.nju.edu.cn)

(南京大学 人工智能学院, 南京 210093)

摘 要: 基于 Aliens 游戏，理解有监督学习中 logistics regression、KNN、random forest 算法，并做出特征提取的优化

关键词: 机器学习；logistics regression 算法；KNN 算法；random forest 算法

中图法分类号: TP301 **文献标识码**: A

1 机器学习方法介绍

以下从 weka 提供的分类模型（classifiers）的其中三种 functions、lazy、trees 类别之中各选择一个，进行介绍，并比较此三种机器学习方法的不同之处以及性能差别

1.1 Logistic Regression 逻辑斯蒂回归（对数几率回归模型）【functions】

逻辑斯蒂回归介绍

回归分析是研究自变量和因变量之间关系的一种预测模型技术并得到基于 input-ouput pairs 的分类模型，同时也是一种常用的统计学方法，经由统计机器学习融入机器学习领域。逻辑斯蒂回归的提出是为了解决单阶跃函数(unit-step function)的函数特性不好的问题（跳跃间断点、定义域上不是全局可微），所以使用对数几率函数(logistic function)$y = \frac{1}{1+e^{-z}}$ 进行近似，也即 sigmoid 函数，将线性模型的结果压缩到[0,1]之间，使其拥有概率意义，它可以将任意输入映射到[0,1]区间，实现值到概率转换。

$$P(y=1|x) = \hat{y} = \sigma(w^T x + b) = \frac{1}{1+e^{-(w^T x+b)}}$$

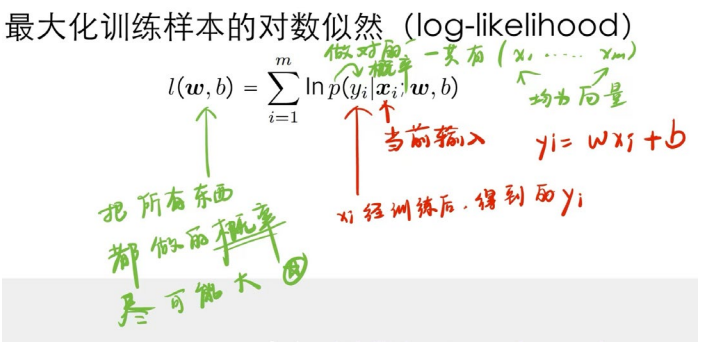
$$P(y=0|x) = 1 - \hat{y} = 1 - \sigma(w^T x + b) = \frac{e^{-(w^T x+b)}}{1+e^{-(w^T x+b)}}$$

所以当广义线性模型 $z = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$ 复合上对数几率函数，即可以得到在概率 0-1 上的连续函数啦，具

体含义为：当 P(y=1|x)的值很接近 0 的时候认为此时 y=0，否则认为 y=1。

逻辑斯蒂回归也可以处理更为复杂的问题，在此类问题之中机器学习的目的是最小化训练误差，即最大化精度，但同时其中的参数是未知的。

此时可以使用极大似然原理 $L(\theta)=L(x_1,x_2,\ldots,x_n;\theta)=\prod_{i=1}^{n}p(x_i;\theta)$. 的变式最大化训练样本的对数似然（极大似然原理的一中应用）



即通过若干次实验，观察其结果，利用结果推出参数的大概值；如果这个参数可以让这个样本出现的概率最大，就把这个参数作为估计的真实值。所以在使用对数几率+广义线性模型+对数极大似然函数可得下式，所以对极大似然函数求极大值，即可得到最理想的参数值。

- **极大似然函数：** $l(\boldsymbol{w},b)=\sum_{i=1}^{m}\ln p(y_i|\boldsymbol{x}_i;\boldsymbol{w},b)$
- **对于对数几率回归模型：**

$$y=\frac{1}{1+e^{-(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b)}}$$

$$p(y=1|\boldsymbol{x})=\frac{e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}$$

$$p(y=0|\boldsymbol{x})=\frac{1}{1+e^{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b}}$$

$$\hat{\boldsymbol{x}}=(\boldsymbol{x};1),\boldsymbol{\beta}=(\boldsymbol{w};b)$$

$$p(y_i|\boldsymbol{x}_i;\boldsymbol{w},b)=y_ip_1(\hat{\boldsymbol{x}}_i;\beta)+(1-y_i)p_0(\hat{\boldsymbol{x}}_i;\beta)$$

$$l(\boldsymbol{\beta})=\sum_{i=1}^{m}(-y_i\boldsymbol{\beta}^{\mathrm{T}}\hat{\boldsymbol{x}}_i+\ln(1+e^{\boldsymbol{\beta}^{\mathrm{T}}\hat{\boldsymbol{x}}}))$$

高阶可导连续凸函数，可以使用梯度下降法、牛顿法求解

逻辑斯蒂回归在 Aliens 中参数的选择和准确性表现

逻辑斯蒂回归中可调节的参数为Ridge value（正则化系数），在test option中选择percentage split选择90%，即 90%的样本用于训练，10%的样本用于测试，可得不同正则化系数对应性能如下

正则化系数	准确率
1e-8（default）	88.2353%
1e-3	90.1961%
1e-2	90.1961%
0.1	88.2353%

由同一组测试数据（在 Train 中跑了三次，并在 weka 中对每个正则化系数完成了 3 次训练，取的平均值）得到的准确率来看，在正则化系数为 0.001 和 0.01 左右的时候效果最好，小于 0.001 的时候（如默认的 1e-8）会导致对训练集的过拟合，无法对测试集达到较好的效果

1.2 KNN（IBK）分类模型【lazy---lazy learning，没有显示的学习过程】

KNN 是一个常见的有监督学习分类模型，通过查询与待检测样本最相近的 k 个训练样本，比较这 k 个样本之中哪个类别的个数更多，此待检测样本就属于哪个类别。此时 K 的大小至关重要

- K 较小—决策更接近数据本身分布，但比较容易受到噪音和异常数据干扰
- K 较大—决策的边界更加平滑，同时对数据变化的敏感程度下降，使泛化误差更小

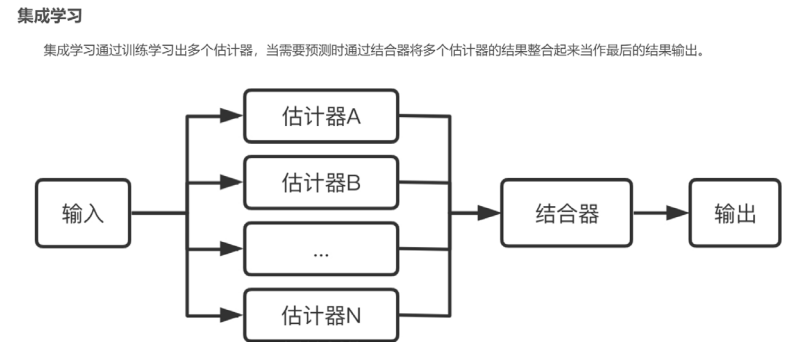
以下为 Aliens 游戏中使用 KNN 进行训练的结果

K	准确率
1（default）	86.2745%
2	88.2353%
3	84.3137%
5	74.4706%
10	72.549%
20	74.4706%

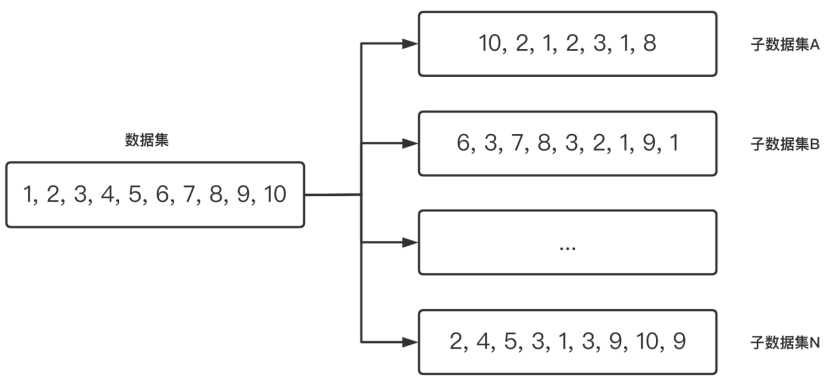
对训练结果有点奇怪，之前使用 KNN 做其他分类任务的时候，多数情况 K 的大小适中的时候效果最好。更换了训练集和测试集的比例之后，训练结果的趋势和上表中保持一致。可能的原因是和 weka 只能把上一局游戏的数据作为训练、测试有关，以及此游戏的特征提取函数会对 K 的选择造成影响。

1.3 随机森林【trees】[1]

随机森林就是通过集成学习的思想将多棵树集成的一种算法，它的基本单元是决策树，而它属于机器学习中的集成学习（Ensemble Learning）方法。随机森林的产生是为了解决决策树泛化能力比较弱的特点，因为一颗决策树的决策流只有一条， 泛化能力弱。而随机森林可以“集思广益”，综合多个决策树做出决策！



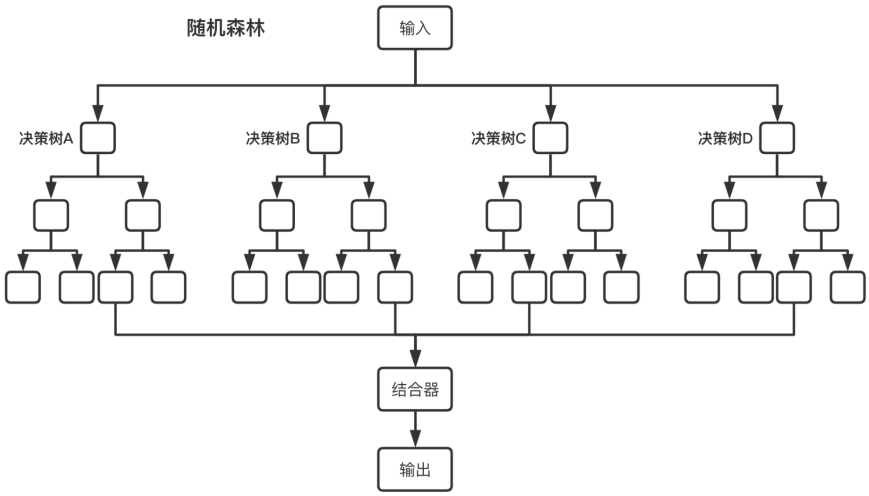
【森林】随机森林是由决策树集成的，每棵决策树都是一个分类器。对于一个输入样本，N 棵树会有 N 个分类结果。而随机森林集成了所有的分类投票结果，将投票次数最多的类别指定为最终的输出，是一种 bagging 的思想，即自助聚集算法（**B**ootstrap **agg**regat**ing**），这和第二种的 KNN 算法一些相似呢！



Boostrap aggregating 算法的具体步骤为：假设有一个大小为 N 的训练数据集，每次从该数据集中有放回的取选出大小为 M 的子数据集，一共选 K 次，根据这 K 个子数据集，训练学习出 K 个模型。当要预测的时候，使用这 K 个模型进行预测，再通过取平均值或者多数分类的方式，得到最后的预测结果。

【随机】一个是随机选取特征，一个是随机样本。比如我们有 N 条数据，每条数据 M 个特征，随机森林会随机 X 条选取样本数据和 Y 个特征，然后组成多个决策树。

综上，随机+森林的解释，随机森林的决策流程如下所示。多颗决策树解决了一颗决策树方差很高的问题，每颗决策树对原数据集进行 bagging 的有放回采样生成，可以保证训练样本的随机性和无差别性。



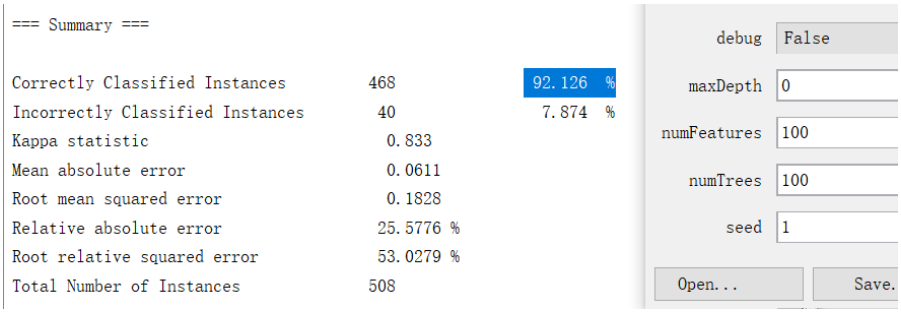
以下为 Aliens 游戏中使用随机森林进行训练的结果

此时有 numFeature、numTrees 和 maxDepth 三个参数， 可对每个变量做对照实验，结果如下。

• 决策树个数过少或过多时候会影响准确率（同 KNN 中 K 的选择），过少-bagging 过程运行次数过少，随机性不足，ballot 的时候效果不好。过多-运行时间太长，跑不出来

• 特征过少或过多，同上分析

• 深度过少和过多，从结果来看，可能每个决策树在深度较低的时候就已经完成了决策（只对 lv1 的地图进行了测试）。所以限制深度过大带来的对训练集的过拟合在 lv1 地图之中并不太适用



一种性能较好的参数如上图所示

2 不同学习方法的性能比对

综上各个学习方法的实验所产生的结果来看，在最优情况下，三者的性能排序如下：

随机森林 > 逻辑斯蒂回归 > KNN

可能的原因如下：

1、 随机森林拥有非线性建模的能力，通过对样本的 bagging，可以产生更多丰富的、随机的样本，可以更好的完成对有限样本信息的挖掘

2、 逻辑斯蒂回归在本例子中表现还可以，其不足的原因可能是特征提取的样本过多，是一个多维向量，在广义线性模型+对数几率中的线性可能对多维向量模拟不佳

```
436 object_at_position_x=19_y=13
437 object_at_position_x=20_y=13
438 object_at_position_x=21_y=13
439 object_at_position_x=22_y=13
440 object_at_position_x=23_y=13
441 object_at_position_x=24_y=13
442 object_at_position_x=25_y=13
443 object_at_position_x=26_y=13
444 object_at_position_x=27_y=13
445 object_at_position_x=28_y=13
446 object_at_position_x=29_y=13
447 object_at_position_x=30_y=13
448 object_at_position_x=31_y=13
```

3、KNN 表现也还可以，但不如前两种方法。KNN 是一种 lazy learning，比较依赖与给的样本以及 K 的选择，前面提到了本例子中 K=2 的时候效果最佳，所以 KNN 所得到的的分类信息会对训练集的拟合过高，当测试集来的时候可能无法准确预测。

3 修改特征提取方法

【修改 1】阅读 featureExtract 代码，发现最后属于 Avatar 的 4 个属性全都存到了一个地方，需要修改

```
feature[448] = obs.getGameTick();
feature[449] = obs.getAvatarSpeed();
feature[450] = obs.getAvatarHealthPoints();//玩家剩余的血量
feature[451] = obs.getAvatarType();
```

在使用当前特征提取进行训练的时候，最有效的消灭 Aliens 的方法思路应该是：

1、尽快让他们在第一行就被消灭掉

2、如果第一行没有被消灭掉，需要移动 Avatar，让它移动到多行 Aliens 叠加的地方，可以一次性消灭多个 Aliens

3、如果遇到了 Aliens 发出的炮弹，需要躲闪

4、Avatar 的移动速度比 Aliens 更快，所以如果当前位置没有打到 Alien，可以移动到 Alien 的前面再次发射炮弹。

但是此时（random forests+最好超参数）Aliens 只能以第一种的思路、很小一部分的第二种和第四种思路进行模拟，但是完全无法躲开 Aliens 发射的炮弹。看 weka 中有关获取的参数的信息，源代码中长度为 453 的 feature 数组其实只能获取当前游戏状态，**但是无法把当前游戏状态的前后状态联系起来。所以训练出来的模型不知道要在炮弹要来的时候移动，也几乎不知道往叠加的 Aliens 和跑道 Aliens 之前来进行攻击**。

445	☐	object_at_position_x=28_y=13
446	☐	object_at_position_x=29_y=13
447	☐	object_at_position_x=30_y=13
448	☐	object_at_position_x=31_y=13
449	☐	GameTick
450	☐	AvatarSpeed
451	☐	AvatarHealthPoints
452	☐	AvatarType
453	☐	class

【修改 2】所以可以增加特征提取的数组大小，新的大小为 N*453，获取 N 个相邻时间内的特征信息。

在 featureExtract 中修改如下，以 3*448+5 作为所有的特征（相邻三个的游戏状态）

```
int N=3;
double[] feature = new double[5+448*N];  // 448 + 4 + 1(class)
```

```
for(int k=0; k<N; k++)
    for(int y=0; y<14; y++)
        for(int x=0; x<32; x++)
            feature[k*448+y*32+x] = map[x][y];
```

```
feature[448*N] = obs.getGameTick();
feature[448*N+1] = obs.getAvatarSpeed();
feature[448*N+2] = obs.getAvatarHealthPoints();//玩家剩余的血量
feature[448*N+3] = obs.getAvatarType();
```

并且同时在 datasetHeader 中进行修改，准备数据集的格式信息

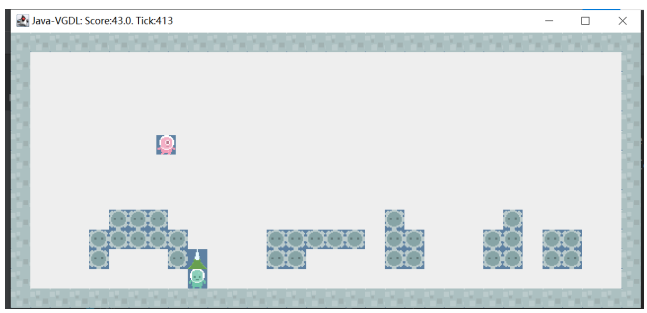
```
public static Instances datasetHeader(){
    FastVector attInfo = new FastVector();
    // 448 locations
    for (int k = 0; k<3; k++){
        for(int y=0; y<14; y++){
            for(int x=0; x<32; x++){
                Attribute att = new Attribute( attributeName: "num of time" + k + "object_at_positi
                attInfo.addElement(att);
            }
        }
    }
```

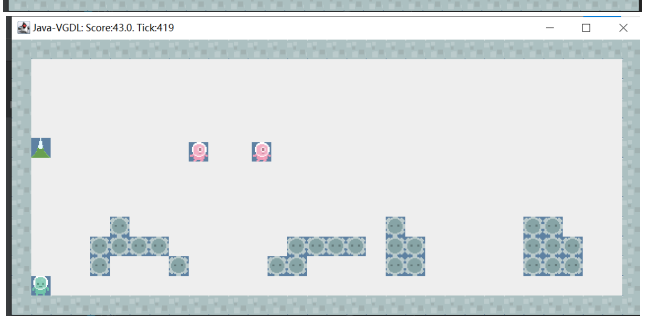
效果如下图所示：在 weka 中的准确率基本上平均增加了百分之 1-百分之 2（有点用，但不多..(。•ˇ‸ˇ•。)...）

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         488               93.3078 %
Incorrectly Classified Instances        35                6.6922 %
Kappa statistic                          0.8637
Mean absolute error                      0.0605
Root mean squared error                  0.1727
Relative absolute error                 24.3969 %
Root relative squared error             49.1381 %
Total Number of Instances              523
```

在实际跑的时候，在多次试验之中，发现了有几次 Avatar 会自己选择收益最高的位置进行攻击





可能的原因可能是收集的数据太少了，weka 只收集上一句的游戏所产生的数据作为训练和测试集，机关 random forest 可以通过 bagging 来丰富输入，但仍是远远不够哒

参考资料

[1] 机器学习算法系列（十八）-随机森林算法（Random Forest Algorithm）

https://blog.csdn.net/sai_simon/article/details/123082619