作业3: Aliens游戏 实验报告

吴政亿 151220129 wuzy.nju@gmail.com
(南京大学 计算机科学与技术系, 南京 210093)

摘要：使用监督学习来模仿人玩游戏的动作，应用朴素贝叶斯分类器，随机森林，AdaBoost与KNN算法建立模型并应用在测试集中。 **关键词**：监督学习、朴素贝叶斯分类器、随机森林算法、AdaBoost算法、KNN算法

1 学习方法介绍

1.1 Naive Bayes

朴素贝叶斯分类器是分类算法集合中基于**贝叶斯理论**的一种算法。它不是单一存在的，而是一个算法家族，在这个算法家族中它们都有共同的规则。例如每个被分类的特征对与其他的特征对都是相互独立的。

朴素贝叶斯基础假设是，对于每一个特征都有：**独立、相等**来支持输出结果。

注意：如果在现实情况中，这个假设就使得朴素贝叶斯不能一般性地正确了。实际上独立这个假设就根本不可能成立，但是又往往在实践中能够很方便地计算。

在进入朴素贝叶斯方程之前，要知道贝叶斯理论是十分重要的。贝叶斯理论指的是，根据一个已发生事件的概率，计算另一个事件的发生概率。贝叶斯理论从数学上的表示可以写成这样：

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

在这里A和B都是事件，$P(B)$不为0。

- 基本上，只要我们给出了事件B为真，那么就能算出事件A发生的概率，事件B也被称为证据。
- $P(A)$是事件A的**先验**（先验概率，例如，在证据之前发生的概率）。证据是一个未知事件的一个属性值（在这里就是事件B）。
- $P(A|B)$是B的后验概率，例如在证据之后发生的概率。

  现在是时候为贝叶斯理论添加假设了，也就是每个特征之间都是相互独立的。所以我们可以将证据分成每个独立的部分。如何两个事件A和B是相互独立的，那么有：

  ```
  $$P(AB)=P(A)P(B)$$
  ```

  因此我们可以得到以下结果：

  ```
  $$P(y|x_1,...,x_n)=\frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$
  ```

  于是又可以写成：

  ```
  $$P(y|x_1,...,x_n)=\frac{P(y)\Pi_{i=1}^{n}P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$
  ```

  因为分母与输入数据是常量相关的，所以我们可以除去这一项：

  ```
  $$P(y|x_1,...,x_n)\propto P(y)\Pi_{i=1}^{n}P(x_i|y)$$
  ```

  现在我们需要建立一个分类模型，我们用已知的类变量yy 的所有可能的值计算概率，并选择输出概率是最大的结果。数学表达式可以这么写：

  ```
  $$\hat{y}=\arg\max_yP(y)\Pi_{i=1}^{n}P(x_i|y)$$
  ```

  所以最后剩下的只有P(y)P(y) 与P(xi|y)P(xi|y) 的计算了。

  请注意：$P(y)$也被称为**类概率**，$P(x_i|y)$ 也被称为**条件概率**。

  不同的朴素贝叶斯分类器差异主要在$P(x_i|y)$ 分布的假设。

- 尽管他们貌似过度简化了假设，朴素贝叶斯分类器在真实世界中的应用还是很不错的，其中著名的文件分类和垃圾邮件过滤就是例子。它只要少量的训练数据就能估计出关键的参数。
- 与其他的复杂方法相比，朴素贝叶斯学习和分类的速度非常快。类条件特征分布的波动意思就是每个分布可以独立地被一个尺寸分布估计出来。这就减轻了维度带来的问题。

1.2 Random Forest

随机森林非常像AdaBoost算法，但区别在于它没有迭代，还有就是森林里的树长度不限制。 因为它是没有迭代过程的，不像AdaBoost那样需要迭代，不断更新每个样本以及子分类器的权重。因此模型相对简单点，不容易出现过拟合。下面先来讲讲它的具体框架流程。

随机森林可以理解为Cart树森林，它是由多个Cart树分类器构成的集成学习模式。其中每个Cart树可以理解为一个议员，它从样本集里面随机有放回的抽取一部分进行训练，这样，多个树分类器就构成了一个训练模型矩阵，可以理解为形成了一个议会吧。然后将要分类的样本带入这一个个树分类器，然后以少数服从多数的原则，表决出这个样本的最终分类类型。

设有N个样本，M个变量(维度)个数，该算法具体流程如下：

1. 确定一个值m，它用来表示每个树分类器选取多少个变量。(注意这也是随机的体现之一)
2. 从数据集中有放回的抽取 k 个样本集，用它们创建 k 个树分类器。另外还伴随生成了 k 个袋外数据，用来后面做检测。
3. 输入待分类样本之后，每个树分类器都会对它进行分类，然后所有分类器按照少数服从多数原则，确定分类结果。

   其中，森林中的每个树越茂盛，分类效果就越好；树和树的枝叶穿插越多，分类效果就越差。

   预选变量个数 (即框架流程中的m)；随机森林中树的个数。这两个参数的调优非常关键，尤其是在做分类或回归的时候。

1.3 AdaBoost

AdaBoost算法是基于Boosting思想的机器学习算法，其中AdaBoost是Adaptive Boosting的缩写，AdaBoost是一种迭代型的算法，其核心思想是针对同一个训练集训练不同的学习算法，即弱学习算法，然后将这些弱学习算法集合起来，构造一个更强的最终学习算法。

为了构造出一个强的学习算法，首先需要选定一个弱学习算法，并利用同一个训练集不断训练弱学习算法，以提升弱学习算法的性能。在AdaBoost算法中，有两个权重，第一个数训练集中每个样本有一个权重，称为样本权重，用向量D表示；另一个是每一个弱学习算法具有一个权重，用向量α表示。假设有n个样本的训练集，初始时$\{(X_1,y_1),(X_2,y_2),\ldots,(X_n,y_n)\}$，设定每个样本的权重是相等的，即$\frac{1}{n}$，利用第一个弱学习算法$h_1$对其进行学习，学习完成后进行错误率$\epsilon$的统计：

$$\epsilon = \frac{\#error}{\#all}$$

其中,$\#error$表示被错误分类的样本数目，$\#all$表示所有样本的数目。这样便可以利用错误率ϵ计算弱学习算法h_1的权重α_1：

$$\alpha_1 = \frac{1}{2}\ln(\frac{1-\epsilon}{\epsilon})$$

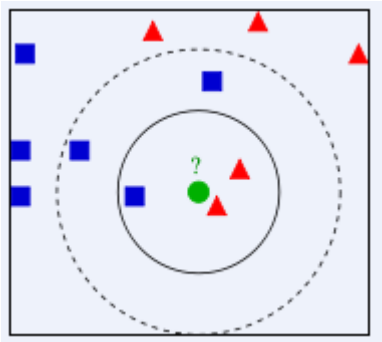
在第一次学习完成后，需要重新调整样本的权重，以使得在第一分类中被错分的样本的权重，使得在接下来的学习中可以重点对其进行学习。

算法总步骤为:

1. 给数据中的每一个样本一个权重
2. 训练数据中的每一个样本，得到第一个分类器
3. 计算该分类器的错误率，根据错误率计算要给分类器分配的权重（注意这里是分类器的权重）
4. 将第一个分类器分错误的样本权重增加，分对的样本权重减小（注意这里是样本的权重）
5. 然后再用新的样本权重训练数据，得到新的分类器，到步骤3
6. 直到步骤3中分类器错误率为0，或者到达迭代次数
7. 将所有弱分类器加权求和，得到分类结果（注意是分类器权重）

1.4 KNN

如下图所示，有两类不同的样本数据，分别用蓝色的小正方形和红色的小三角形表示，而图正中间的那个绿色的圆所标示的数据则是待分类的数据。也就是说，现在， 我们不知道中间那个绿色的数据是从属于哪一类（蓝色小正方形or红色小三角形），下面，我们就要解决这个问题：给这个绿色的圆分类。　　我们常说，物以类聚，人以群分，判别一个人是一个什么样品质特征的人，常常可以从他/她身边的朋友入手，所谓观其友，而识其人。我们不是要判别上图中那个绿色的圆是属于哪一类数据么，好说，从它的邻居下手。但一次性看多少个邻居呢？从下图中，你还能看到：



- 如果K=3，绿色圆点的最近的3个邻居是2个红色小三角形和1个蓝色小正方形，少数从属于多数，基于统计的方法，判定绿色的这个待分类点属于红色的三角形一类。
- 如果K=5，绿色圆点的最近的5个邻居是2个红色三角形和3个蓝色的正方形，还是少数从属于多数，基于统计的方法，判定绿色的这个待分类点属于蓝色的正方形一类。

于此我们看到，当无法判定当前待分类点是从属于已知分类中的哪一类时，我们可以依据统计学的理论看它所处的位置特征，衡量它周围邻居的权重，而把它归为(或分配)到权重更大的那一类。这就是K近邻算法的核心思想。

> KNN算法中，所选择的邻居都是已经正确分类的对象。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。
>
> KNN 算法本身简单有效，它是一种 lazy-learning 算法，分类器不需要使用训练集进行训练，训练时间复杂度为0。KNN 分类的计算复杂度和训练集中的文档数目成正比，也就是说，如果训练集中文档总数为 n，那么 KNN 的分类时间复杂度为O(n)。
>
> [KNN](https://baike.baidu.com/item/KNN/3479559)方法虽然从原理上也依赖于极限定理，但在类别决策时，只与极少量的相邻样本有关。由于KNN方法主要靠周围有限的邻近的样本，而不是靠判别类域的方法来确定所属类别的，因此对于类域的交叉或重叠较多的待分样本集来说，KNN方法较其他方法更为适合。
>
> K 近邻算法使用的模型实际上对应于对特征空间的划分。K 值的选择，距离度量和分类决策规则是该算法的三个基本要素：

1. K 值的选择会对算法的结果产生重大影响。K值较小意味着只有与输入实例较近的训练实例才会对预测结果起作用，但容易发生过拟合；如果 K 值较大，优点是可以减少学习的估计误差，但缺点是学习的近似误差增大，这时与输入实例较远的训练实例也会对预测起作用，使预测发生错误。在实际应用中，K 值一般选择一个较小的数值，通常采用交叉验证的方法来选择最优的 K 值。随着训练实例数目趋向于无穷和 K=1 时，误差率不会超过贝叶斯误差率的2倍，如果K也趋向于无穷，则误差率趋向于贝叶斯误差率。
2. 该算法中的分类决策规则往往是多数表决，即由输入实例的 K 个最临近的训练实例中的多数类决定输入实例的类别

3. 距离度量一般采用 Lp 距离，当p=2时，即为欧氏距离，在度量之前，应该将每个属性的值规范化，这样有助于防止具有较大初始值域的属性比具有较小初始值域的属性的权重过大。

   KNN算法不仅可以用于分类，还可以用于回归。通过找出一个样本的k个最近邻居，将这些邻居的属性的平均值赋给该样本，就可以得到该样本的属性。更有用的方法是将不同距离的邻居对该样本产生的影响给予不同的权值(weight)，如权值与距离成反比。　该算法在分类时有个主要的不足是，当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的K个邻居中大容量类的样本占多数。 该算法只计算“最近的”邻居样本，某一类的样本数量很大，那么或者这类样本并不接近目标样本，或者这类样本很靠近目标样本。无论怎样，数量并不能影响运行结果。可以采用权值的方法（和该样本距离小的邻居权值大）来改进。

   > 该方法的另一个不足之处是计算量较大，因为对每一个待分类的文本都要计算它到全体已知样本的距离，才能求得它的K个最近邻点。目前常用的解决方法是事先对已知[样本点](https://baike.baidu.com/item/%E6%A0%B7%E6%9C%AC%E7%82%B9)进行剪辑，事先去除对分类作用不大的样本。该算法比较适用于[样本容量](https://baike.baidu.com/item/%E6%A0%B7%E6%9C%AC%E5%AE%B9%E9%87%8F)比较大的类域的自动分类，而那些样本容量较小的类域采用这种算法比较容易产生误分。
   >
   > 实现 K 近邻算法时，主要考虑的问题是如何对训练数据进行快速 K 近邻搜索，这在特征空间维数大及训练数据容量大时非常必要。

2 原有特征提取方法结果

2.1 Naive Bayes

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.677     0.072      0.838     0.677     0.749      0.94     0
                 0.897     0.073      0.947     0.897     0.921      0.961    1
                 1         0.031      0.479     1         0.648      0.996    2
                 1         0.071      0.266     1         0.42       0.991    3
Weighted Avg.    0.824     0.072      0.878     0.824     0.84       0.956

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 201  25  21  50 |   a = 0
  39 444   4   8 |   b = 1
   0   0  23   0 |   c = 2
   0   0   0  21 |   d = 3
```

Naive Bayes学习了一直保持发射子弹的动作，没有学习到开局移动到中央的左移动作，另外，Naive Bayes并不能很好的学习好我给的训练集中对于下落子弹的规避动作，在子弹下落时无意义的移动。另外他几乎无法学习在后期针对最底下的怪物进行移动击杀的动作。

2.2 Random Forest

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.955     0.028      0.94      0.955     0.948      0.997    0
                 0.988     0.011      0.99      0.988     0.989      1        1
                 0.946     0.008      0.921     0.946     0.933      0.999    2
                 0.891     0.001      0.98      0.891     0.933      0.999    3
Weighted Avg.    0.967     0.015      0.967     0.967     0.967      0.999

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 235   4   6   1 |   a = 0
   5 409   0   0 |   b = 1
   4   0  70   0 |   c = 2
   6   0   0  49 |   d = 3
```

Random Forest中AI习得了大部分的动作，如一直保持发射子弹，子弹坠落时移动避开，在只剩部分怪物时向怪物移动击杀等。但是对于我开局移到中间方便左右移动的动作过拟合导致开局直接移动到最左侧。

2.3 AdaBoost

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.955     0.534      0.448     0.955     0.61       0.711    0
                 0.577     0.067      0.905     0.577     0.705      0.755    1
                 0         0          0         0         0          0.58     2
                 0         0          0         0         0          0.68     3
Weighted Avg.    0.601     0.201      0.615     0.601     0.56       0.72

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 235  11   0   0 |   a = 0
 175 239   0   0 |   b = 1
  60  14   0   0 |   c = 2
  55   0   0   0 |   d = 3
```

通过观察可以发现，AdaBoost除了一直发射子弹外，其他都没有学习到.....

2.4 KNN

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.972     0.035      0.926     0.972     0.948      0.997    0
                 0.986     0.008      0.993     0.986     0.989      1        1
                 0.919     0.006      0.944     0.919     0.932      0.999    2
                 0.873     0          1         0.873     0.932      0.999    3
Weighted Avg.    0.967     0.016      0.968     0.967     0.967      0.999

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 239   3   4   0 |   a = 0
   6 408   0   0 |   b = 1
   6   0  68   0 |   c = 2
   7   0   0  48 |   d = 3
```

KNN的效果十分不错，开局移动到中央持续子弹射击，在有子弹坠落时规避，在最后几个怪物时移动击杀，但是在最后这个特征，也许是特征不够，他并不会按照最下面优先级最高进行击杀，而且追击时移动的方向也有些问题。

2.5 四种方法对比

	Naive Bayes	Random Forest	AdaBoost	KNN
开局调整位置	F	T(一半)	F	T
连续射击子弹	T	T	T	T
子弹规避	T(一半)	T	F	T
优先击杀底层	F	T	F	T(一半)

3 特征提取改进

首先，对于源代码的特征提取方案，有一处有误,在源代码中他们都存在448中。下面是更正的代码。

```
// 4 states
feature[448] = obs.getGameTick();
feature[449] = obs.getAvatarSpeed();
feature[450] = obs.getAvatarHealthPoints();
feature[451] = obs.getAvatarType();
```

然后，我将原有的记录所有NPC位置变为只记录我所在的一列与他左右两个邻居列的NPC，并且每次记录我左边与右边的NPC数量。因为在游戏中，我们只会规避自己所在的一列以及周围的炮弹，并且会根据敌人所在的位置移动。在这里我列出一部分在`Recorder.java`中的改动。

3.1 public static double[] featureExtract(StateObservation obs)

```
if( obs.getNPCPositions()!=null )
    for(ArrayList<Observation> l : obs.getNPCPositions()){
        for(Observation o:l){
            int NPC_X=(int)(o.position.x/25);
            if(NPC_X<avatarPos_X)
                left_NPC_number++;
            else if(NPC_X>avatarPos_X)
                right_NPC_number++;
        }
        allobj.addAll(l);
    }

for(Observation o : allobj){
    Vector2d p = o.position;
    int x = (int)(p.x/25);
    int y= (int)(p.y/25);
    if(Math.abs(x-avatarPos_X)<=1)
        map[x-avatarPos_X+1][y] = o.itype;
    if(o.itype==5 && Math.abs(p.x-avatarPos.x)<25)//被炸弹炸到
        hasBomb=true;
}
//left and right NPC number
feature[452]=left_NPC_number;
feature[453]=right_NPC_number;
feature[454] = hasBomb ? 10.0 : 0.0;
```

3.2 public static Instances datasetHeader()

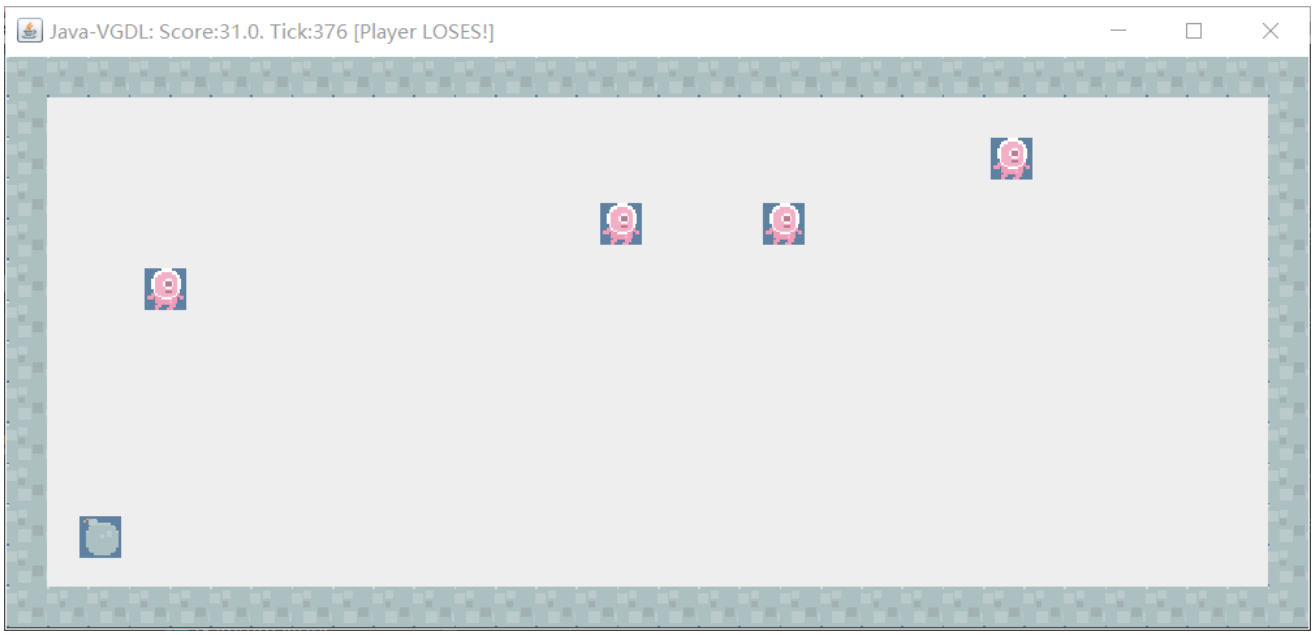
```
//left and right NPC number
att = new Attribute("left_NPC_number");attInfo.addElement(att);
att = new Attribute("right_NPC_number");attInfo.addElement(att);
att = new Attribute("hasBomb" ); attInfo.addElement(att);
```

4 修改特征函数后性能对比

4.1 Naive Bayes

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class

                 0.791     0.063      0.822     0.791     0.806      0.961    0
```

```
                 0.965     0.135      0.941     0.965     0.953      0.985    1
                 0.455     0.017      0.556     0.455     0.5        0.933    2
                 0         0          0         0         0          ?        3
Weighted Avg.    0.896     0.11       0.892     0.896     0.894      0.977

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 106  20   8   0 |   a = 0
  12 333   0   0 |   b = 1
  11   1  10   0 |   c = 2
   0   0   0   0 |   d = 3
```



在改进后，他准确的习得了开局移到中间的操作与击杀最底层怪物的行为，但是他对于躲避子弹表现的还不够好。

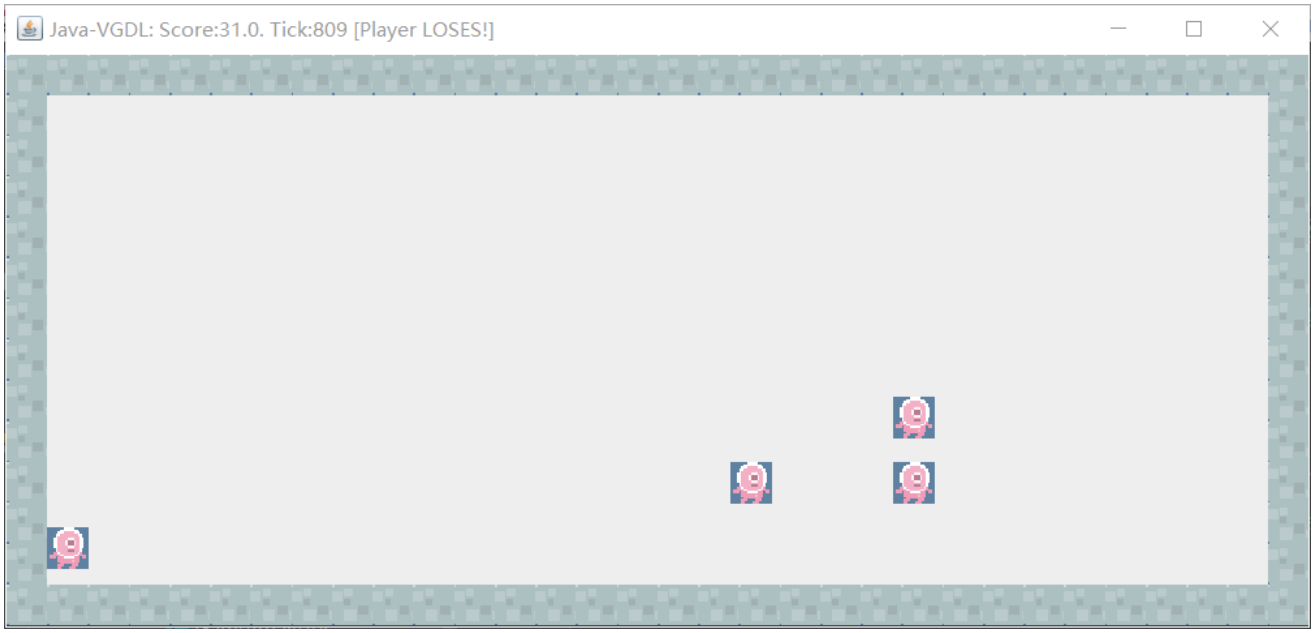
4.2 Random Forest

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0          1         1         1          1        0
                 1         0          1         1         1          1        1
                 1         0          1         1         1          1        2
                 0         0          0         0         0          ?        3
Weighted Avg.    1         0          1         1         1          1

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 134   0   0   0 |   a = 0
   0 345   0   0 |   b = 1
   0   0  22   0 |   c = 2
   0   0   0   0 |   d = 3
```

Random Forest中AI习得了部分的动作，如一直保持发射子弹，开局移动到中间进行射击。但是他在子弹坠落时不懂得移动避开。

4.3 AdaBoost

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0.144      0.717     1         0.835      0.943    0
                 0.91      0          1         0.91      0.953      0.973    1
                 0         0          0         0         0          0.849    2
                 0         0          0         0         0          ?        3
Weighted Avg.    0.894     0.039      0.88      0.894     0.88       0.959

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 134   0   0   0 |   a = 0
  31 314   0   0 |   b = 1
  22   0   0   0 |   c = 2
   0   0   0   0 |   d = 3
```



通过观察可以发现，AdaBoost习得了开局移动位置，规避子弹，追击最底层的怪物，但是在只剩下少量怪物时，他只追击而不射击，估计是将射击动作当成了噪音……

4.4 KNN

```
=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0          1         1         1          1        0
                 1         0          1         1         1          1        1
                 1         0          1         1         1          1        2
                 0         0          0         0         0          ?        3
Weighted Avg.    1         0          1         1         1          1

=== Confusion Matrix ===

   a   b   c   d   <-- classified as
 134   0   0   0 |   a = 0
   0 345   0   0 |   b = 1
   0   0  22   0 |   c = 2
   0   0   0   0 |   d = 3
```

KNN的效果十分优秀，开局移动到中央持续子弹射击，在有子弹坠落时规避，在最后几个怪物时移动追击进行击杀。

4.5 四种方法对比

	Naive Bayes	Random Forest	AdaBoost	KNN
开局调整位置	T	T	T	T
连续射击子弹	T	T	T(一半)	T
子弹规避	F	F	T	T
优先击杀底层	T	F	T	T

可见在更改特征函数后，Naive Bayes，AdaBoost，KNN的表现均有不同层次的提升，而Random Forest却微微下降，从理论上来讲，我们得到的结果应该是上升的，然而事实上对于Random Forest来说实验结果不尽如人意，这其中的误差可能来自以下几个方面：

1. 样本数目过少，因为Random Forest最大的优势体现在大样本上，小样本准确性可能并不会很好。
2. 训练集中对于子弹规避与优先击杀底层体现的左右操作不够明显

引用

1. https://www.cnblogs.com/muchen/p/6883263.html
2. https://blog.csdn.net/sinat_36246371/article/details/60140664
3. https://blog.csdn.net/google19890102/article/details/46376603
4. https://blog.csdn.net/lx85416281/article/details/40656877
5. https://www.jianshu.com/p/a6426f4c4e64
6. https://baike.baidu.com/item/k近邻算法/9512781?fr=aladdin