This agent is developed based on classification. The training data was collected by you through the pervious human labelling experiment. In particular, experienced Minecraft players watch playing video and give an assessment for the performance.

There are three levels: 0 (beginner), 1 (mediate player), 2 (advance player). Using this data, we train and test a classifier.

Here are some details:

We used the cross validation for getting all the results.

1. The ground truth data and prediction results by the random forests (the best classifier we found):

Please notice, ‘human’ represents the label provided by human, which is the ground truth; and ‘pred’ is the label predicted by the model.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Label 0 (human | pred) | Label 1 (human | pred) | Label 2 (human | pred) |
| way1: only majority vote | 146 | 145 | 137 | 155 | 57 | 40 |
| way2: delete multi-major | 100 | 87 | 107 | 136 | 53 | 37 |
| way3: evaluate multi-major | 130 | 137 | 1. | 150 | 77 | 53 |

Because in the raw data, each video was labelled by multiple users (3-5 users). So, we first combine labels of each video using majority vote. However, as there is not always a winner in the majority vote. We handle the data (videos) that does not have a winner in different ways.

In ‘way2’, we delete all the videos which does not have a winner in the majority vote. In ‘way1’, we select a lower label for those videos (for instance, when four labels of a video are 0,0,1,1; we select label 0 for this video). In ‘way 3’, we select a label based on the confidence of users (for instance, when four labels of a video are 0,0,1,1; and the average confidence of users give label 1 is higher, we select label 1 for this video).

According to the above table, the best results are from ‘way 2’. In this case, the exact prediction accuracy is:

#total used data = 276

Summary:

precision recall f1-score support

0.0 0.60 0.52 0.56 2000

1.0 0.52 0.66 0.58 2140

2.0 0.70 0.49 0.58 1060

accuracy 0.57 5200

macro avg 0.61 0.56 0.57 5200

weighted avg 0.59 0.57 0.57 5200

accuracy of predicting (0 or 1) vs 2: **85.4%**

accuracy of predicting neighbor classes (1 to 2, 2 to1, 2 to 3 or 3 to 2) or correct classes: **96.15%**

confusion matrix:

[[1040 860 100]

[ 600 1420 120]

[ 100 440 520]]

way1 is also good in overall, but relatively worse in predicting label 2:

#total used data = 341

Summary:

precision recall f1-score support

0.0 0.62 0.62 0.62 2920

1.0 0.53 0.60 0.56 2740

2.0 0.57 0.40 0.47 1140

accuracy 0.57 6800

macro avg 0.57 0.54 0.55 6800

weighted avg 0.58 0.57 0.57 6800

accuracy of predicting (0 or 1) vs 2: 85%

accuracy of predicting neighbour classes (1 to 2, 2 to1, 2 to 3 or 3 to 2) or correct classes: 95%

confusion matrix:

[[1800 1020 100]

[ 860 1640 240]

[ 240 440 460]]

Way3 didn’t improved the accuracy as we supposed (in fact it is the worst), probably because the “confidence” of different participants is hard to measure in the same standard. Some people tend to be confident with their conclusion while some other people are not.

Notice:

* This is a multiple class classification task, which is harder than binary classification. In principle, a random guess accuracy for this task shall be 33.3%.
* To further evaluate our prediction results, we check the wrongly predicted videos and would like to confirm what kinds of misclassification our model did. We found that most misclassifications are among nearby classes (for instance, predict a video with label 0 to label 1, or predict a video with label 1 to label 2, verse vice). In summary, videos classified in the neighbour classes (1 to 2, 2 to1, 2 to 3 or 3 to 2) or correct classes account for over 90% of all videos.

Feature engineering:

What features we use in the end? All 20 features in the list?

We use 20 features, 0-2 + 5-19 in the description list. For feature 3 & 4,

Feature 3: 1st gain time sequence with only rewardable items. The reward assigning table is also in description file. For items that cannot get reward, set the correspond index to be “0”; the rest indexes are the same with feature 1.

Feature4: 1st gain order sequence with only rewardable items. For items that cannot get reward, set the correspond index to be “0”; the rest indexes are the same with feature 0.

(motivation: the non-rewardable items are kind of useless, so when and which order the player got them might be less importance. But we are not fully sure, so we remain both feature 0,1 and 4,3, let the classifier to do the decision. In fact, the importance shows that they have same level of importance.)

How we generate the features?

For feature 5-19, we can get it directly from the input data.

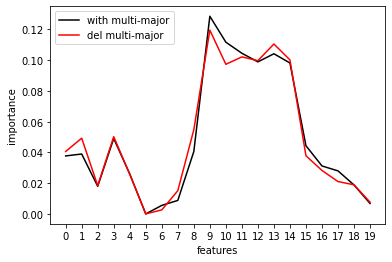
For feature 0-4:

Firstly we implement kmeans algorithm on all training data for each feature 0-4 and got 3 clusters. So each training data for each feature 0-4 has a label (0,1, or 2). We save the 5 models, mainly to save the cluster centers. When training the RF model, we directly use the assigned labels; when predicting a new episode data, we put each of the sequential features (0-4) into the correspond k-means model and get the label, then put into the RF model.

For developers:

The 20 features in the list are required, but for feature 3 & 4, it is ok if we give empty list [].

Other notes:



From the Feature important figure, we can see that the No.9-14(such as dense reward, attack ratio) feature have the highest importance, and No. 5-6, No.19 have the lowest importance.

Others:

File ‘data\_optim.npy’ includes the cleaned training data for way 2.

File ‘data\_origin.npy’ includes the cleaned training data for way 1.