E-181 : Final Project

Due on Thursday, May 8, 2014

Pac-Man Ghost Hunting

Praphruetpong Athiwaratkun

Overview

This project has two major components. The first one is to train machine learning algorithms to distinguish ghosts and capsules using the data collected. Once the algorithms are trained, Pac-Man can use the ability to recognize ghosts and capsules to help determine the optimal actions through the game.

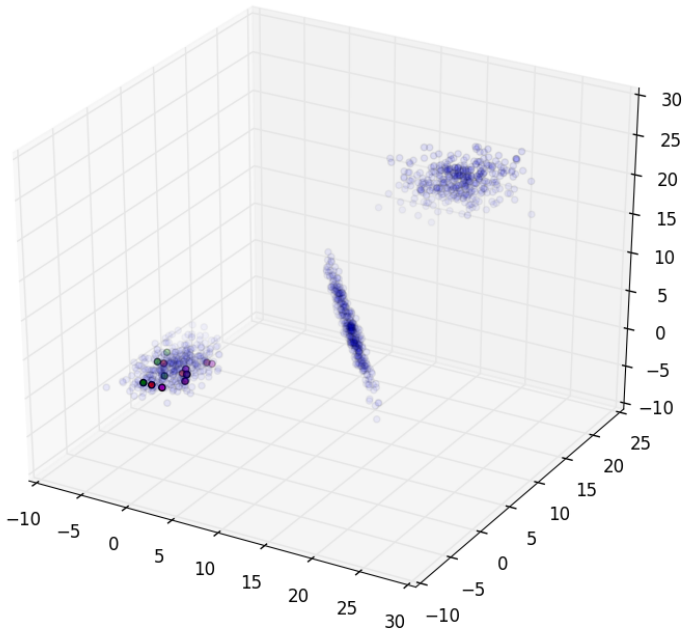
Training

Since it is very helpful for Pacman to be able to identify good capsules, bad ghosts, and predict the scores of each good ghost, a lot of emphasis is put on this section to make sure the training parameters will result in accurate predictions. The data are obtained from running `ExampleTeam` agent. The size of data set for the number of ghost features is $76,052$. The size of the data set for the number of capsule features is $1,170$.

Classification - Capsules

Note that the capsule data obtained from the data collection phase are unlabeled. To help decide what machine learning algorithm to use, I plot of the training data which are not labeled and the data from `getGoodCapsuleExamples()` from various number of games (with different seeds).

Figure 1: Capsule Features (Transparent Blue) Versus Good Capsule Features from Various Games



K Means Algorithm

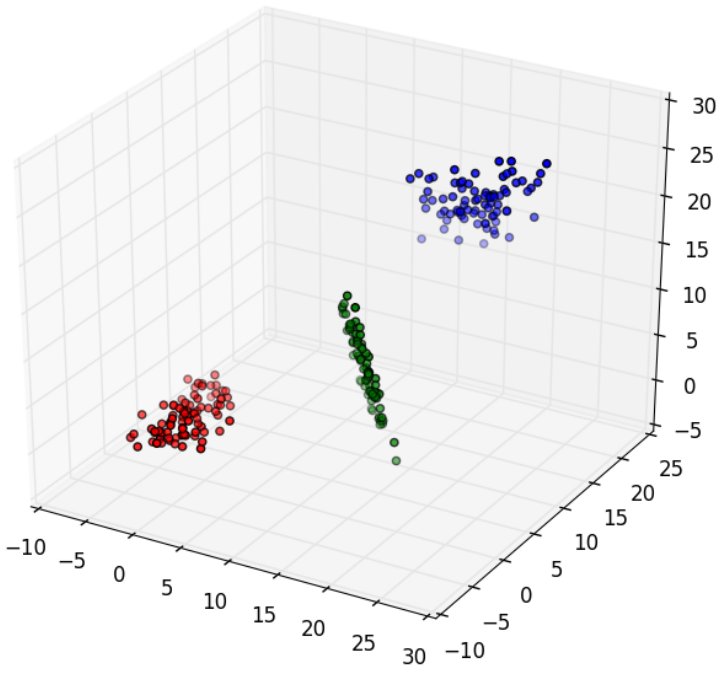
The data comprise of three distinct clusters, one of which clearly belongs to non-placebo capsules. To classify the capsules, I decide to use Kmeans algorithm with the number of clusters being 3. I use the Kmeans class

from $scikit - learn$ where the training is initialized by KMeans++. Note that we could have chosen the number of clusters to be 2 if we know for sure that the good capsules will always be in one cluster, as in Figure 1.

Prediction

Unsurprisingly, K Means does exceptionally well in classifying the 3 clusters since they are well separated. In the plot below, features with the same color have the same classified results from K Means.

Figure 2: Classification By K-Means



After training, I use the learned parameters to predict good capsules by:

- At the start of any given game, obtain the good capsule features from `getGoodCapsuleExamples()`.
- Classify the good capsule features and calculate the mode of the classified labels. Use this mode as the label of good capsules.
- During the game, classify if a capsule is good by predicting the label with Kmeans and see if the label is the same as the label of the good capsules.

Classification - Ghosts

In this step, we aim to train a machine learning algorithm to be able to classify classes of ghosts. Note that the valid ghost labels are $0, 1, 2, 3$ and 5. However, I choose to group the good ghosts as one class for simplicity. Note that the distinction among good ghosts as to which class they're in can be helpful for predicting scores but there is no loss of generality to group all the good ghosts together if we simply want to distinguish the bad ghosts from the rest.

Algorithm

Since the features are quite high dimensional, I choose to use a soft-margin SVM with a non-linear kernel for the classification. The "regularization" parameter $C = 1$. The kernel is given by

$$K(x, x') = e^{-\frac{1}{13}|\vec{x}-\vec{x'}|_2^2}$$

Results

Below is the result from $5-$fold cross validation.

Table 1: Ghost Classification Accuracy by SVM

Iteration	Cross Validation Scores
1	0.97580698
2	0.97508382
3	0.97764629
4	0.97587114
5	0.9756739
Average	**0.976016426**

Regression - Ghost Scores

Knowing the scores of good ghosts can be helpful in determining which ghost to hunt. I decide to use linear regression and see how accurate the prediction is. The scores could very well be non-linear on the features but we can always improve the accuracy with more advanced algorithm if need be.

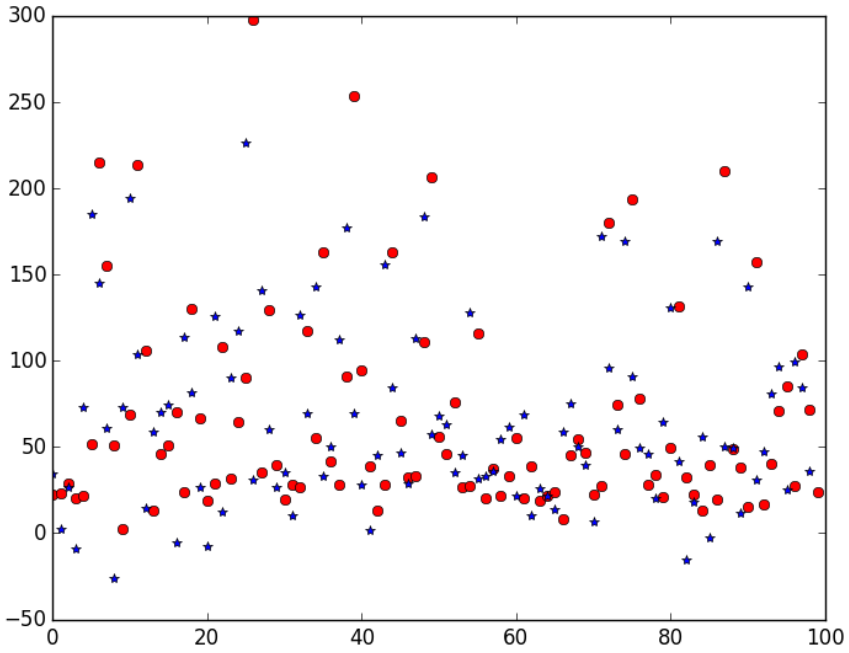
Results

Table 2: Score Prediction by Linear Regression (Coefficient R^2)

Iteration	Cross Validation Scores
1	0.85852589
2	0.8620412
3	0.8590795
4	0.85959122
5	0.86454468
Average	**0.860756498**

Training continued on next page...

Figure 3: Predicted Scores(Blue) Versus Training Data (Red)



Later in the game, I find that fine-tuning other parts of Pac Man's algorithm seems to be more important that being able to correctly predict the ghosts' juiciness precisely.

Pacman's Strategy

As of now, we have reasonably accurate algorithms to distinguish good capsules, bad ghosts, and estimate ghosts' scores. The next task is to determine how Pacman can leverage this information and use it for the game in real-time.

Approach

We can potentially use Q-learning for Pacman. However, the state space will be quite large and the transitions are complicated. My approach is to define methods that do each task optimally and combine them with some adjusted parameters to form Pac-Man optimal strategy.

Strategy

Below is a high level summary of the steps taken to implement Pacman's decision.

- If there's any scared ghost and we're able to locate a bad ghost, hunt it.
- If there's at least one bad ghost and the distance from the bad ghost to Pac Man is less than some adjustable parameter γ, then set on Pacman on the capsule hunting mode. (This means that the capsule hunting persists if the distance become less that γ after stepping.)
- In the rare case that there is no bad ghost that can be identified, hunt a capsule and avoid all ghosts. This happens roughly 3% according to the cross validation error.

Implementation Details

Hunting Good Capsule

While hunting a good capsule, Pac Man will always avoid a bad ghost. In particular, Pac Man does not take an action that results in the Pac Man's successor position being directly north, south, east, or west of the bad ghost.

Optimally Travelling to An Item

To travel to each item, after filtering out the actions that are not allowed (such as wall or bad ghost's adjacent spot), Pac Man minimizes `self.distancer.getDistance( Pac Man Position, Item Position)` over the possible actions left. If there are ties, then Pac Man chooses an action that minimizes the Euclidean distance. This implementation has proven to work well especially in hunting the bad ghosts who always tries to escape when it is scared.
Note that we did not use the score prediction to influence Pacman's decision. If we are to do so, there will be another flexibility to adjust whether Pac Man should hunt a ghost with a given score based on the distance.

Determining γ

The parameter γ corresponds to the distance between Pac Man and the identified bad ghost that causes Pac Man to hunt a capsule and subsequently hunt a bad ghost. If γ is large, then Pac Man always tries to find a capsule when there is no scared ghost. If γ is small, Pac Man waits until the bad ghost comes near him before setting off to find a capsule and in the mean time escapes from being hunted. It is not clear whether Pac Man should keep hunting the good ghosts and wait until a bad ghost approaches. Note that if Pac Man waits long enough, the bad ghost naturally walks towards the Pac Man to hunt him. Waiting around could save Pac Man time and hence boost the scores, given that Pac Man only has 1000 steps to walk.
I run Pac Man with multiple seeds and multiple values of γ to determine the optimal one.

Pacman's Strategy continued on next page...

Table 3: Pac Man Scores

Seed / Distance	6	8	10	13	16	21	∞
10	18231	28239	15989	15501	17239	24614	17646
15	35177	34442	32964	33371	36991	36991	36991
2	26934	25751	21372	19156	19883	22113	25155
20	18786	14448	21607	19068	22552	19349	20436
3	23066	18246	28697	21450	22578	23281	23281
45	23555	30047	24235	37053	33949	33949	33949
5	16989	20323	16276	10420	-2741	11910	11910
51	18079	32420	23863	25908	19554	21030	26962
53	27710	22338	21719	25499	27232	25794	28554
58	24080	21145	22387	28047	26950	23552	23552
7	10126	14171	12047	15420	13185	13504	17017
87	16004	13454	12547	10518	14733	19808	19808
90	22951	14323	24868	13230	28818	19114	13234
Average	21668	22257	21428	21126	21609	22693	**22961**

From the result in Table 3, $\gamma = \infty$ yields the highest average score. Therefore, the submitted code for Pac Man has γ set to ∞. However, we can see there are a lot of variations among different games. Note that the reported scores for each distance is quite noisy as it highly depends on the ghost's behavior while escaping from Pac Man, the locations of the good capsules which are influenced by the capsules Pac Man previously ate, etc.