Obstacles Avoidance with Machine Learning Control Methods in Flappy Birds Setting

Yi Shu*[1], Ludong Sun†[1], Miao Yan‡[1], and Zhijie Zhu§[1]

[1]Department of Mechanical Engineering, Stanford Univerisity

December 12, 2014

Abstract

Object avoidance is an important topic in control theory. Various traditional control methods can be applied to achieve control of object path such as PID, Bang-Bang control, sliding mode control. Given a known and simple dynamic system, those classical controls method work pretty well. However, controls of complex, multi degree of freedom systems or controls of systems with unknown dynamics have been pushing the limit of traditional control laws. This report adopts machining learning methods of Support Vector Machine(SVM) with linear kernels and reinforcement learning using value iteration to solve control problems in the game 'Flappy Bird' without understanding the dynamics of the problem. For comparison purposes, Bang-Bang control is also implemented and described in the report. The game is also modified to increase difficulty for further comparison.

The machine learning methods are shown to significantly improve the results got from Bang-Bang Control. In the modified game version with moving pipes, Reinforcement Learning is more feasible than SVM. Detailed implementation of and comparison among each methods are discussed in this report.

Keywords:Machine Learning, Controls, Obstacles Avoidance

*yishu@stanford.edu
†ldsun@stanford.edu
‡miaoy@stanford.edu
§zhuzj@stanford.edu

1 Introduction

With the increasing popularity of autonomous vehicle, unmanned aerial vehicle, humanoid robots and etc., new methods of controlling complex system to avoid object are points of interests among control engineers. Many classical control methods are based on accurate models or concrete knowledge to derive dynamics of systems and corresponding control laws. For complicated or unknown dynamics systems, classical control methods are not very efficient and new systematic methods are needed to solve such problems. Machine learning draws much attention in last decades and is well applied in many areas such as shopping recommendation systems, computer vision, speech recognitions and etc. Applying Machine learning to solve complicated control problems is investigated in this paper.

Games are ideal environments for developing, implementing and testing Machine Learning algorithm. "Flappy Bird" system is chosen to compare control methods and machine learning methods in general and also compare different machine learning algorithms.

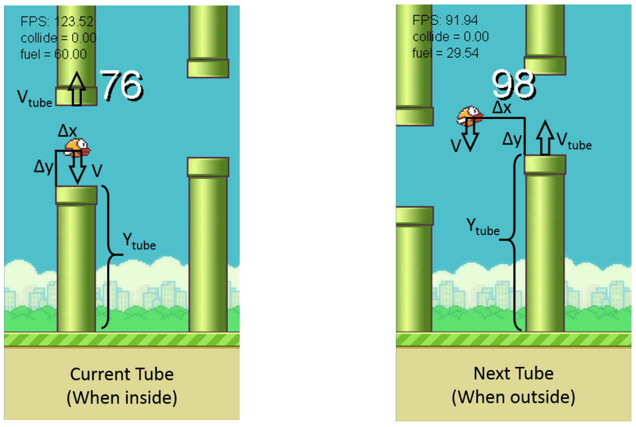
The game "Flappy Bird" is a side-scrolling mobile game, which was a very popular in early 2014. The objective is to direct a flying bird, named "Faby", which moves continuously to the right between sets of pipes. If "Faby" touches the pipes, the game ends. Each time the player taps the screen, the bird briefly flaps upwards; if the screen is not tapped, the bird falls to the ground due to gravity, which also ends the game. The game has all necessary features: a moving object and obstacles. The game itself is repetitive, hard and addictive. The goal is to achieve autonomous flying of birds travelling between pipes. When and whether to flap is determined by control methods and Machine Learning algorithms.

2 Problem Formulation

There are two different scenarios. The first scenario is the standard Flappy Bird. For the second one, to raise the difficulty of the game, the pipes are changed to be able to move upwards and downwards at a specific speed.

The decision will be based on the relative and absolute parameters of the pipes and the bird. As Fig.1 shows, there is a multi-dimensional attribute space consisting of:

- V - bird speed in vertical direction
- Δx - axis distance relative to the front of next pipe
- Δy - axis distance relative to the bottom of next pipe
- V_{pipe} - Pipes' velocity (modified version)
- Y_{pipe} - Pipes' position (modified version)



(a) First Situation (b) Second Siutation

Figure 1: Bird's feature selection

3 Proposed Solutions

It is natural to implement the SVM and Reinforcement Learning. This is a classification problem with discretizable continuous features, so SVM is chosen. When the complexity of the task increases, for instance, the obstacle is moving in a random pattern, it will be hard for human to supervise the learning process and to generate training sets. Thus Reinforcement Learning is applied as an implicit supervised learning algorithm.

3.1 Bang-Bang control

From years' experience in classical control, by analyzing the game's source code, we consider that Bang-Bang control is one of the most effective classical control methods that comes to our mind and expect Bang-Bang control to at least help birds go through a few tubes. So just for comparison purpose, simple Bang-Bang control is implemented by using only one feature Δy. The bird only takes action on the basis whether it is above the tip of next pipe.

3.2 SVM

The SVM is applied in the first scenario, so the first three of attributes are considered. The training set of the problem is not linearly separable in the feature space of raw data. The SVM maps the input vectors into high-dimensional feature space and gives the maximum margin hyper plane. SVM with a linear kernel is adopted and *liblinear library*[2] is used to implement SVM algorithm. The problem is constructed in the following way.

$$min_{\gamma,w,b}\frac{1}{2}\left||w|\right|^2$$

$$s.t.\quad y^{(i)T}\left(w^Tx^{(i)}+b\right)\geq 1, i=1...n$$

Data:

The SVM is implemented in a static pipe setting. For simplicity, only the first pipe ahead of the bird is taken into consideration.

The training set is generated by several trials of human. If human gives the command to flap, the current state is labeled as 1, otherwise current state is labeled as -1. At every sampling step, a current state $x^{(i)}=[\Delta x,\Delta y,V]$ and related label $y^{(i)}\in\{-1,1\}$ are recorded in the training set.

As illustrated in Fig.2, the training set generated by human in the original state space is not linearly separable. In Fig.2, red crosses and circles represent training points labeled with flapping and not flapping respectively. Thus, a high-dimension feature space is chosen. Nine features from fundamental analysis are selected. The dimension of the data is increased by adding second and third order of the feature. In Fig.2, blue crosses and circles represent testing result labeled with flapping and not flapping respectively.

$$\Phi=\begin{bmatrix}\Delta x & \Delta y & V & \Delta x^2 & \Delta y^2 & V^2 & \Delta x^3 & \Delta y^3 & V^3\end{bmatrix}$$

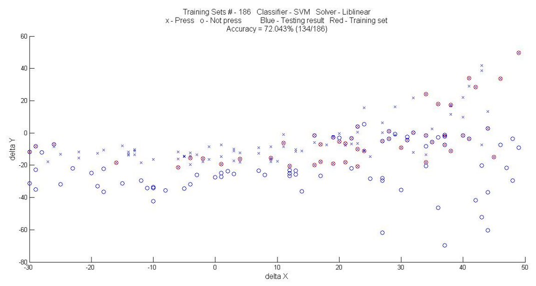


Figure 2: SVM Training Result

$$k(x, z) = \Phi^T \Phi$$

3.3 Reinforcement Learning

The Markov decision processes (MDP) settings of the reinforcement learning will be presented in details below.

A. States

There are four states contributing to the control of the bird: $\Delta x, \Delta y, V_{b\_p}, y_p$.

Definitions of Δx and Δy are the same as mentioned in last section of SVM. $V_{b\_p} = V_b - V_p$ denotes the vertical velocity difference between the bird and the pipe, because the movement of the pipe is also considered in this section. y_p denotes the pipe's position in y direction.

In the fixed obstacle case, state vector $s = [\Delta x, \Delta y, V_{b\_p}]$ is applied, which is chosen by the intuition of the specific physical system. In the moving obstacle case, state vector $s = [\Delta x, \Delta y, V_{b\_p}, y_p]$ is applied. The reason why y_p is added is that the movement of pipes is nonlinear. The y position of the pipes has upper and lower boundaries, so that a feasible solution for the bird's trajectory exists. The pipe will reverse when reaching the upper or lower boundary, so y_p will be necessary for the learning algorithm to determine future movement of the pipe.

Due to the program implementation of the game, $\Delta x, \Delta y, V_{b\_p}, y_p$ are naturally discretized. However, due to 'the curse of dimensionality', the computation load for reinforcement learning is incredibly large with the four dimension state space. Thus, for computational cost concerns, the state space is divided to coarser grids. The discretization is illustrated in Fig.3 below. Areas further from the pipe gap are discretized with coarser grids, since the control decision is more natural in the area.

B. Action

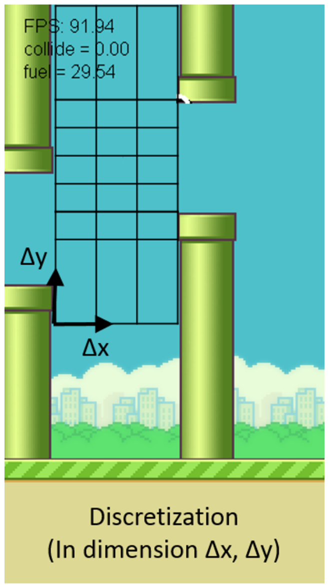


Figure 3: RL State Discrete

Action taken at every state is binary: flap or not flap, represented by $a = 1$ and $a = 0$ in the program. At every time step the program will read the current state of the bird and generate an action based on the learning result from last iteration.

C. Rewards

At every iteration, if the bird is in collision with a pipe, we set the reward of last state:

$$R_t(s, a) = -100, t = T$$

All the states before collision are rewarded with:

$$R_t(s, a) = 1, t = 1, 2, \ldots, T-1$$

The total reward for state s and action a is updated at every time step:

$$R_{t+1}^{total}(s, a) = R_t^{total}(s, a) + R_t(s, a)$$

After every iteration is terminated by the collision, the actual reward for every state is calculated by (if the state has been visited):

$$R(s, a) = \frac{\text{total reward in state } s \text{ with action } a}{\text{\#times we took action } a \text{ in state } s}$$

D. State Transition

According to the known dynamics of bird, it seems that $P_{sa}(s')$ denoting the state transition probability from state s to s' with action a is deterministic and can be computed explicitly as a priori. However, due to the switching of target pipe when the bird is going out from last pipe and checking for a new pipe, the state transition at this point is not 'continuous' as the case of targeting the same pipe. Randomness appears in the state transition because the relative position of the new pipe is unpredictable given the state relative to last pipe. Thus, we are unable to compute transition probability matrix P_{sa} with only the knowledge of bird dynamics. P_{sa} is computed 'on line' during every iteration of reinforcement learning, according to the equation below: $P_{sa}(s') =$

$$\frac{\#\text{times we took action } a \text{ from state s to state } s'}{\#\text{times we took action } a \text{ in state } s}$$

With increasing quantity of data collected in iterations, $P_{sa}(s')$ will finally converge to the real value.

E. Optimization Algorithm

Due to discretization, the size of state space is limited to the level of two thousand. For small MDPs, value iteration is often very fast and converges with a small number of iterations. Thus, we use value iteration to find the value function.

The value iteration is processed as follows:
For each state, initialize $V(s) := 0$.
Repeat until converge
{ For every state, update:

$$V(s) = R(s) + \max_{a \in A} \gamma \Sigma_{s'} P_{sa}(s')V(s')$$

}

$\gamma$is set 0.95 which reflects the effect of future actions on current decision.

After several iterations, $V(s)$ and P_{sa} are well learnt by the algorithm. To apply the control law given by the learning algorithm given a state s, the control input is given by:

$$a^* = \max_{a \in A} P_{sa}(s')V(s)$$

4 Simulation Results and Disscusion

The criterion to compare different methods is the number of pipes the birds can fly through before the end of the game. The x axis is the number of iteration the game played. The y axis is logarithm of the times the birds fly through pipes successfully in each round.

Fig.4 shows the results when the standard game is played. The methods are compared. The blue line and pink line show the average results of Bang-Bang Control and SVM, respectively. Moving average of Reinforcement Learning results are shown in the red line. It is clear that Machine Learning methods generate better results than the classical control law. In a short period (less than two hours), SVM is significantly better than reinforcement learning. The video link-1 is the game played under SVM. It is worth noting that there is an up-going trend of results of Reinforcement Learning. It has been proven that Reinforcement Learning can help birds perform better than SVM (Video-Link 2).

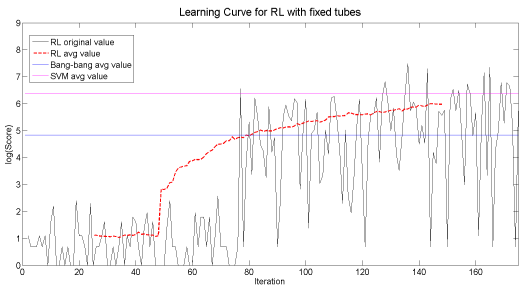


Figure 4: Learning Process of Reinforcement Learning for fixed pipes

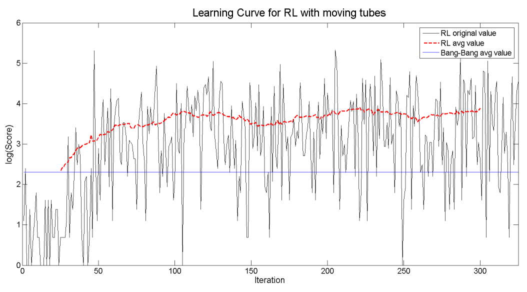


Figure 5: Learning Process of Reinforcement Learning for moving pipes

Fig.5 shows the results of modified game. The pipes are moving randomly. The game is so hard that scoring more than five is very challenging for humans. In this case, Bang-Bang control still works but with lower scores. SVM is not feasible due to the lack of enough data. The powerful of reinforcement learning comes into play. Reinforcement learning is obviously better than Bang-Bang Control. After training overnight, it looks like moving pipe is not a big problem and the results got from reinforcement learning in moving case are as good as the standard case. Link-3 shows the game (moving pipe case) was played under reinforcement learning. The reason why reinforcement learning performs well is that the policy we generated can determine the right action in a specific state, which is hard to find using traditional control law.

Video Link-1 (SVM in standard game)

`https://www.youtube.com/watch?v=cYFeI9eFaBY`

Video Link-2 (Reinforcement Learning in standard game)

`https://www.youtube.com/watch?v=UwfnUNhkcCg`

Video Link-3 (Reinforcement Learning in modified game)

`https://www.youtube.com/watch?v=UM9B2iHIMCw`

5 Future Development

In the evaluation of reinforcement learning, it has been found that a more refined space of data and longer training time will enhance the performance of the learning algorithm. With limitation of laptop, the full potential of a refined data space and long training time has yet to be explored.

However, when discretization of the states is refined, data storage and iteration speed become issues. A possible way to resolve this might be adopting sparse matrix to store state transition data in that some states do not transit to 'remote' states. Another option is to find the smart way to discrete the states in the reinforcement learning, e.g.,applying sampling-based method to extract states from the original continuous space.

Another future improvement of our work is incorporating more complex dynamic models of the birds to the learning algorithm. The reinforcement learning deals with the control of the bird without knowing the dynamics of the bird. However, the dynamics of the 'bird' in the game is merely a simplified version of projectile motion. In reality, the 'bird' might be a real vehicle that has various available control inputs and a constraint of fuel consumption. The vehicle can decide to ascend, descend, accelerate and decelerate. Besides, uncertainty of the environment and dynamic system, such as wind disturbance and modeling error of the dynamics, also affect the decision model in the learning algorithm. These factors might bring new challenges to the current implementation of reinforcement learning algorithm.

6 Conclusion

This report focused on applying Machine Learning to solve control related problems. The game 'Flappy Bird' is used to test different algorithms. Distance in x and y, bird's velocity were chosen as standard features. For the modified version adding moving pipes, two addition features (pipes' velocity and pipes' position) were added. Bang-Bang control, SVM and Reinforcement Learning were applied to both standard game and modified Flappy Bird with moving tubes. It turns out that Machine Learning is immensely useful for controlling complicated system or system without known dynamics. Comparing between SVM and Reinforcement Learning, for fixed pipes, the SVM and reinforcement learning generally perform similarly. SVM is much better because it has a much higher data space. For moving pipes, it is almost not possible for human to play smoothly and generate enough data so SVM is not feasible due to the lack of training data. Reinforcement learning still works very well.

To be clear, Table 1 below summarizes the advantages and disadvantages of each methods.

Table 1: Comparsions of three methods

	Bang-Bang Control	**SVM**	**Reinforcement Learning**
PRO	Easy to implement; Fast Computing	Relatively less computing time; work well in standard situations	Robust in different circumstances
CON	Inaccurate; Limited	Un-robust in complicated situation	Expensive computing; Long training period

7 Acknowledgement

We want to thank Professor Ng and CS229 TAs for teaching this wonderful course. We learned a lot in this quarter. We also would like to thank Mingjing Zhang for providing Flappy Bird source code so that we can modify and add Machine Learning components into it.

References

[1] Andrew Ng *"CS229 Machine Learning Course Materials", [online] 2014, http://cs229.stanford.edu/materials.html* 2014.

[2] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. *LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008), 1871-1874. Software available at http://www.csie.ntu.edu.tw/ cjlin/liblinear*