Lifelong Learning: A case study of punting balls

Ravikiran Janardhana
Department of Computer Science
University of North Carolina at Chapel Hill
Email: ravikirn@cs.unc.edu

***Abstract*—Designing robots that learn by themselves to perform complex real-world tasks is a still-open challenge for the field of Robotics and Artificial Intelligence. In this project, I use the case study of punting balls as a lifelong learning problem. The robot is fed with demonstrations of punting a variety of balls and records the target distanced acheived and updates the physics model of the punt with each demonstration. The learning process is continuous and not stagnant and as a result of this, the internal model needs to be bounded by a finite memory and we need to represent model/data in an efficient format. After the initial demonstrations, the robot is able to predict the launch velocity and the angle of elevation given any new ball and a target distance to reach. The efficient data representation of the model is achieved using the combination of Gaussian Mixture Models (GMM) and Gaussian Mixture Regression (GMR) and I show that the memory required to store the such an internal model compared to storing each of the individual demonstrations is substantially small. Using this physics model, the robot can predict the configuration to punt a new ball to reach a target distance with high accuracy.**

I. Introduction

Throughout the last decades, the field of robotics has produced a large variety of approaches to automate and perform a variety of tasks. Despite significant progress in virtually all aspects of robotics science, most of today's robots are specialized to perform a narrow set of tasks in a very particular kind of environment. Most robots employ specialized controllers that have been carefully designed by hand, using extensive knowledge of the robot, its environment and its task. If one is interested in building autonomous multi-purpose robots, such approaches face some serious bottlenecks such as knowledge bottleneck (not aware of all the dynamics at design time), engineering bottleneck (most models are explictly hand coded) and tractability bottleneck (high computational complexity).

Machine learning aims to overcome these limitations, by enabling a robot to collect its knowledge on-the-fly, through realworld experimentation. If a robot is placed in a novel, unknown environment, or faced with a novel tak for which no a priori solution is known, a robot that learns can collect new experiences, acquire new skills, and eventually perform new tasks all by itself. For example, in [1] a robot manipulator is described which learns to insert a peg in a hole without prior knowledge regarding the manipulator or the hole. Maes and Brooks [2] successfully applied learning techniques to coordinating leg motion for an insect-like robot. Their approach, too, operates in the absence of a model of the dynamics of the system. Learning techniques have frequently come to bear in situations where the physical world is extremely hard to model by hand (e.g., the characteristics of noisy sensors). For example, Pomerleau describes a computer system that learns to steer a vehicle driving at 55mph on public highways, based on input sensor data from a video camera [3]. Learning techniques have also successfully been applied to speed-up robot control, by observing the statistical regularities of "typical" situations (like typical robot and environment configurations), and compiling more compact controllers for the frequently encountered. For example, Mitchell [4] describes an approach in which a mobile robot becomes increasingly reactive, by using observations to compile fast rules out of a database of domain knowledge.

However, there is a principle shortcoming in most of to date's rigorous learning approaches. Most of the robot control learning approaches focus on learning to achieve single, isolated performance tasks. If one is interested in learning with a minimum amount of initial knowledge, as is often the case in approaches to robot learning, such approaches have a critical limiting factor the number of training examples required for successful generalization. The more complex the task at hand and the lesser is known about the problem beforehand, the more training data is necessary to achieve the task. In many robotics domains the collection of training data is an expensive undertaking due to the slowness of robotics hardware. Hence, it does not surprise that the time required for real-world experimentation has frequently been found to be the limiting factor that prevents rigorous machine learning techniques from being truly successful in robotics.

The task of learning from scratch can be significantly simplified by considering robots that face whole collections of control learning problems over their entire lifetime. In such a lifelong robot learning scenario [5], [6], learning tasks are related in that they all play in the same environment, and that they involve the the same robot hardware. Lifelong learning scenarios open the opportunity for the transfer of knowledge across tasks. Complex tasks, which might require huge amounts of training data when faced in isolation, can conceivably be achieved much faster if a robot manages to exploit previously learned knowledge. For example, a lifelong learning robot might acquire general-purpose knowledge about itself and its environment, or acquire generally useful skills that can be applied in the context of multiple tasks. Such functions, once learned, can be applied to speed up learning in new tasks.

In order to transfer knowledge across various tasks, first we need to have an efficient physical model representation

of any task at hand rather than storing each of the individual demonstration as is. Also, it is expected that the robot continues to learn via new demonstrations and this places a limit on the amount of data it can store. In this project, I have used the example of punting a ball and analyzed the physical model required to replicate the punt trajectories. I present a novel method to store the punt physical model extracted using Gaussian Mixture Models (GMM) in an efficient matrix structure which can be later used to retrieve trajectories using Gaussian Mixture Regresstion given any new configuration. By doing so, I show that the memory required to store the model using my method is substantially smaller when compared to storing all of the trajectory data. I also show that the retrieved trajectory from this model is highly accurate without losing too much of information.

II. Method

A. Physics model of a punt

Equations for hang time and horizontal distance can be derived from the projectile equations of motion

$$y(t) = v_y t - \frac{1}{2} a_y t^2 \tag{1}$$

$$x(t) = v_x t \tag{2}$$

where y and x are the height and horizontal displacements respectively, v_y and v_x are its vertical and horizontal velocities, a_y is its vertical acceleration, and t is its time in flight. An expression for hang time, T, results from solving Eqn 1 for t when y = 0 and a = 9.8m/s^2 ,

$$T = \frac{2 v_0 sin(\theta_0)}{9.8} \tag{3}$$

where v_0 and θ_0 are initial velocity and launch angle respectively.

However, the above equations neglect air resistance which can lead to a substantial error when dealing in a practical environment. Air resistance, or drag, is a viscous force in a direction opposing the velocity of the projectile. Air drag, W, is given by the relation,

$$W = \frac{1}{2} \rho C_D A v^2 \tag{4}$$

where ρ is the air density, C_D is the drag coefficient, A is the cross-sectional area of the projectile normal to the trajectory and v is the speed of the projectile relative to the air.

In order to include the drag into the physics model, the acceleration in x and y directions have to be suitably modified as below,

$$a_x = -C_D v v_x \tag{5}$$

$$a_y = -C_D v v_y - g \tag{6}$$

where g = 9.8m/s^2 and the negative sign indicates that these forces are acting against the launch velocity.

Consider a scenario where the robot needs to predict the launch velocity and angle so that a football reaches a target distance of 100 meters assuming the Drag Coefficient C_D in air is 0.006. Table I shows the variation of maximum horizontal distance (max_x) and total time taken (t_{total}) by the projectile with and without taking drag forces into account. Using the learned model, the robot now predicts the correct launch velocity and angle to reach the target distance using a greedy optimizer. Figure 1 shows the pictorial representation of the scenario described by Table I.

TABLE I
VARIATION OF MODEL PARAMETERS

Model	v_0	θ_0	max_x	t_{total}
Without Drag	31.3 m/s	45°	100.01 m	4.52 secs
With Drag	31.3 m/s	45°	63.21 m	3.6 secs
With Drag Corrected	43.31 m/s	44°	100.01 m	4.7 secs

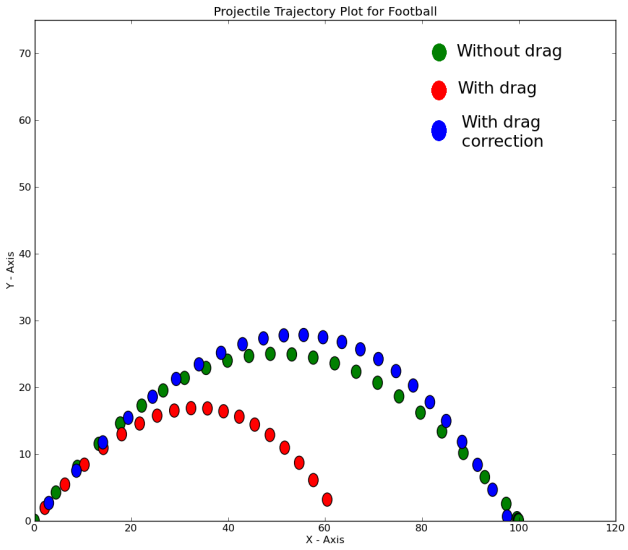

Fig. 1. Projectile with and without drag

III. Conclusion

The conclusion goes here.

Acknowledgment

The authors would like to thank...

References

[1] Vijaykumar Gullapalli, Judy A. Franklin, and Hamid Benbrahim. *Acquiring robot skills via reinforcement learning. IEEE Control Systems*, i72(1708): 13-24, February 1994.

[2] Pattie Maes and Rodney A. Brooks. *Learning to coordinate behaviors.* In Proceedings Eighth National Conferenceon Artificial Intelligence, pages 796-802, Cambridge, MA, 1990. AAAI, The MIT Press.

[3] D. A. Pomerleau. *ALVINN: an autonomous land vehicle in a neural network.* Technical Report CMU-CS-89- 107, Computer Science Dept. Camegie Mellon University, Pittsburgh PA, 1989.

[4] Tom M. Mitchell. *Becoming increasingly reactive.* In Proceedings of1990 AAAl Conference, Menlo Park, CA, August 1990. AAAI, AAAl Press I The MIT Press.

[5] Sebastian B. Thrun and Tom M. Mitchell. *Lifelong robot learning. Robotics and Autonomous Systems, 1993.* Also appeared as Technical Report IAI-TR-93-7. University of Bonn, Dept. of Computer Science 111.

[6] Thrun, S. B. 1994. *A Lifelong Learning Perspective for Mobile Robot Control.* In Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems, 23-30. Washington, D.C.: IEEE Computer Society.