International Journal of Advanced Smart Convergence Vol.7 No.3 15-22 (2018)
http://dx.doi.org/10.7236/IJASC.2018.7.3.15

IJASC 18-3-3

Control of Single Propeller Pendulum with Supervised Machine Learning Algorithm

TserendondogTengis* and Amar Batmunkh*

Mongolian University of Science and Technology, Ulaanbaatar, Mongolia
E-mail:{tengis, abatmunkh }@must.edu.mn

Abstract

Nowadays multiple control methods are used in robot control systems. A model, predictor or error estimator is often used as feedback controller to control a robot. While robots have become more and more intensive with algorithms capable to acquiring independent knowledge from raw data. This paper represents experimental results of real time machine learning control that does not require explicit knowledge about the plant. The controller can be applied on a broad range of tasks with different dynamic characteristics. We tested our controller on the balancing problem of a single propeller pendulum. Experimental results show that the use of a supervised machine learning algorithm in a single propeller pendulum allows the stable swing of a given angle.

Keywords: *cost, gradient, control, data, propeller, regression*

1. Introduction

Since ancient times, people want to build a machine capable of thinking. Since the concept of artificial intelligence began, it has become a major area of research. Machine learning (ML) is a field of computer science that gives computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed [1]. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome [2]. Modern definition of ML is: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" [3, 4].

In general, any machine learning problem can be assigned to one of two broad classifications:

- Supervised learning
- Unsupervised learning

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs [5, 6]. Supervised learning problems are categorized into "regression" and "classification" problems [7]. In a regression problem, we are trying to predict results within a continuous

Manuscript Received: June. 20, 2018 / Revised: June. 26, 2018, 2018 / Accepted: July. 2, 2018
Corresponding Author: tengis@must.edu.mn
Tel:+976-99095618,
Dept. of Electronic Engineering, Mongolian University of Science and Technology, Mongolia

output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

Unsupervised machine learning is the machine learning task of inferring a function to describe hidden structure from "unlabeled" data [4].

In the previous research, we have successfully tested the PID and State space feedback control to maintain the stability of a single propeller pendulum [8-21]. In order to apply the above control methods, it is necessary to define the dynamic and kinematic models of the system.

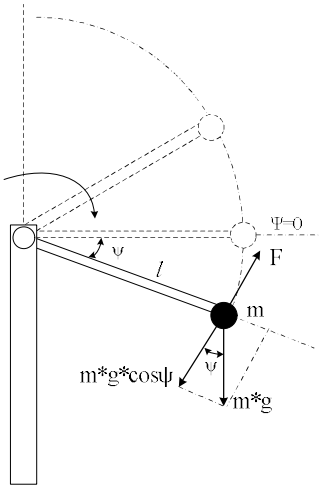

Figure 1. Propeller based pendulum model

In Fig. 1 shown the experimental model, where l represents distance of brushless motor from pivot center and ψ represents Euler angle about x body axis, m is the total mass of brushless dc motor with propeller that is at the end of the lever, g is the acceleration due to gravity. F represents thrust force produced by brushless DC powered propeller motor mounted at the end of lever. The pendulum is balanced against gravity in a single direction and upright balancing is not possible. After applied voltage, the propeller spins and generates torque to pull up the pendulum. The torque is caused a sum of these forces tangential components to the rotating multiplied with corresponding distances from the pivot point. Neglecting the friction force and the effect of body moments on the translational dynamics, an expression of forces acting on the pendulum according to Newton's laws is derived as:

$$-lF + lmg\cos\psi = -l\ddot{\psi}lm \tag{1}$$

Here, $\ddot{\psi}$ is angular acceleration.

In this research work we does not use the dynamic and the kinematic models of the object. Instead using the input and output values of this system we build real-time control system based on supervised machine learning algorithm and tested on the microcontroller. Real model of the propeller based pendulum is shown in Fig. 2.

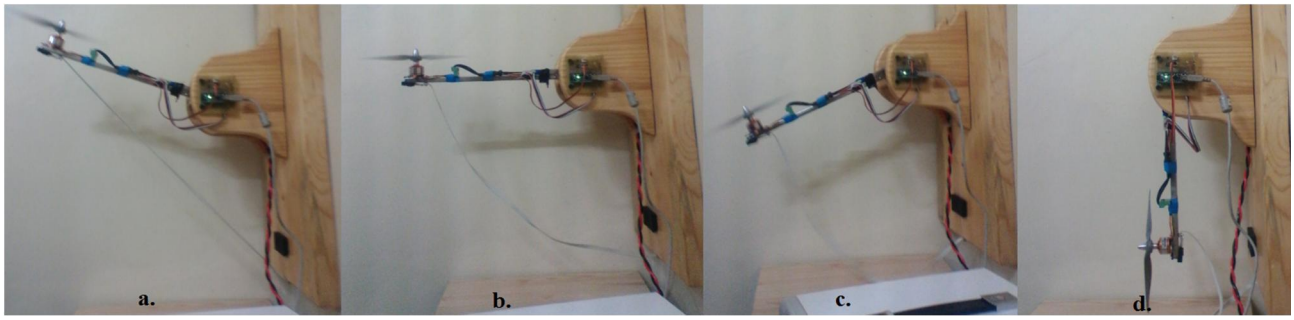

Figure 2. Propeller based pendulum real model (a. +10° angle, b. 0° angle, c. -45° angle, and d. -90° angle stability)

2. Supervised machine learning method

The main idea of machine learning is to evaluate the function from collected training data [4]. The purpose of our study is to maintain the stability of a single propeller pendulum on certain angle. To do this, we collected data by increase speed of the motor by PWM signal and generate lifting force. The control variables for this system is the angle of the pendulum with horizontal axis and the manipulating variable is rotation speed given to motorized propeller. Collected data are shown in Table 1.

Table 1. Training data

m	x (Angle)	y (PWM)	m	x (Angle)	y (PWM)
1	-84.36	760	16	-69.26	850
2	-86.41	766	17	-67.14	856
3	-86.72	772	18	-64.51	862
4	-87.4	778	19	-59.92	868
5	-88.34	784	20	-57.22	874
6	-88.64	790	21	-54.76	880
7	-87.52	796	22	-51.06	886
8	-86.82	802	23	-46.15	892
9	-84.35	808	24	-42.99	898
10	-83.3	814	25	-39.4	904
11	-81.41	820	26	-32.27	910
12	-77.88	826	27	-27.05	916
13	-76.51	832	28	-22.4	922
14	-74.67	838	29	-16.65	928
15	-73.28	844	30	0	934

We use $x^{(i)}$ to denote the "input" variables (in our case is angle), also called input features, and $y^{(i)}$ to denote the "output" or target variable that we are trying to predict (PWM). A pair $(x^{(i)}, y^{(i)})$ is training example, and the dataset that we will be using to learn a list of m training examples $(x^{(i)}, y^{(i)})$; i=1,…., m-is called a training set. For example $(x^{(1)}, y^{(1)}) = (-84.36, 760)$. Graphical presentation of the relation is shown in Fig. 3a.

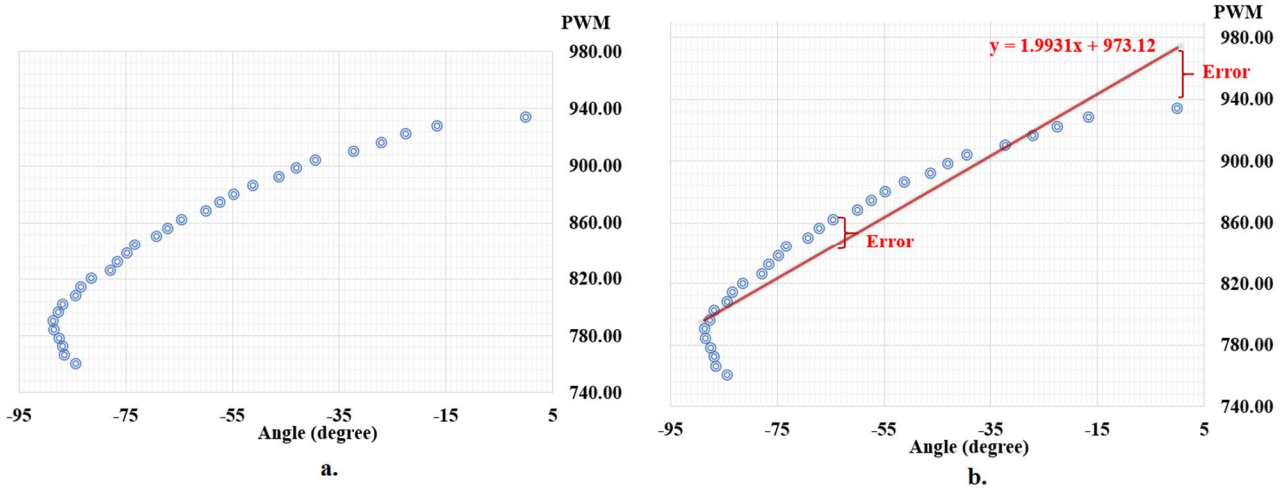

Figure 3. a. Collected data and b. linear regression

In the previous research, we have tested linear regression model (Fig. 3b). Line function has been identified and was observed that some of the angles did not reach or they exceeded the reference angle.

3. Defining the Cost function

To describe the supervised learning problem, our goal is, given a training set, to learn a function $h: X \rightarrow Y$ so that $h(x)$ is a "good" predictor for the corresponding value of y. This function, h, is called a hypothesis. Learning process shown in Fig. 4.

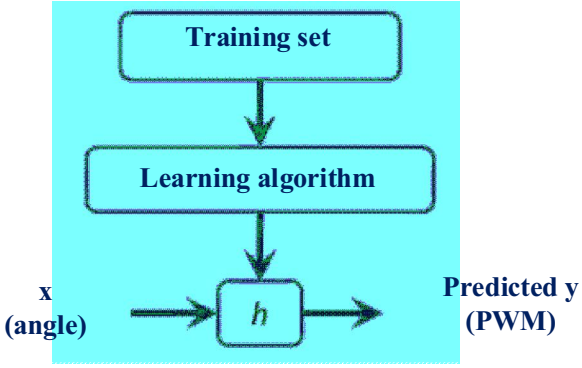

Figure 4. Learning process

The form of our selected hypothetic function is second order polynomial function (2). Functions of a higher order could be chosen. In this case, the relative linearity may be violated due to deviation in collected data.

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \quad (2)$$

Here, θ_i are parameters.

The idea is we need to choose the parameters $\theta_0, \theta_1, \theta_2$ so that $h(x)$ meaning the value we predict on input x that this is at least close to the values y for the examples in our training set. In other words we want that the difference between the predicted values of $h(x)$ and the actual value y to be small.

$$h_\theta(x^{(i)}) - y^i$$

We are trying to make a curve which passes through these scattered data points in Figure 3a. The best

possible curve will be such that the average squared vertical distances of the scattered points from the line will be the least. We can measure the accuracy of our hypothesis function by using a **cost function.** This takes an average difference of all the results of the hypothesis with inputs from *x*'s and the actual output *y*'s.

$$J(\theta_0, \theta_1, \theta_2) = \frac{1}{2m}\sum_i^m (h_\theta(x^{(i)}) - y^{(i)})^2 \quad (3)$$

Formula (3) is the cost function and we want to do is minimize over $\theta_0, \theta_1, \theta_2$.

$$\min_{\theta_0\,\theta_1\,\theta_2} J(\theta_0, \theta_{1,}\theta_2) \quad (4)$$

4. Parameter learning with Gradient descent

We have hypothesis function (2) and we have cost function (3) that measuring how well it fits into the data. Now we need to estimate the parameters $\theta_0, \theta_1, \theta_2$ in the hypothesis function using gradient descent method. When cost function is at the very bottom of the pits in graph, then its value is the minimum and we estimated the optimal parameters.

The way we do this is by taking the derivative of cost function. The slope of the tangent is the derivative at that point and it will give us a direction to move towards. We make steps down the cost function in the direction with the steepest descent. The size of each step is determined by the parameter α, which is called the learning rate. We start some value of $\theta_0, \theta_1, \theta_2$ keep changing $\theta_0, \theta_1, \theta_2$ to reduce $J(\theta_0, \theta_1, \theta_2)$ until to end up at a minimum.

Repeat until convergence ()

$$\{\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \theta_2)\} \quad (5)$$

for(j=0, j=1 and j =2)

We must simultaneously update $\theta_0, \theta_1, \theta_2$.

for (;;)

{

$$t\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1, \theta_{2,})$$
$$t\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1, \theta_{2,})$$
$$t\theta_2 := \theta_2 - \alpha \frac{\partial}{\partial \theta_2} J(\theta_0, \theta_1, \theta_{2,})$$
$$\theta_0 := t\theta_0$$
$$\theta_1 := t\theta_1$$
$$\theta_2 := t\theta_2 \quad (6)$$

}

If α is too small gradient descent will be small and will take lot of loops in controller to reach the minimum value. If α is very large the gradient descent can overshoot the minimum value and divert.

To find minimum value of parameters we need take partial derivatives from cost function.

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \theta_2) = \frac{\partial}{\partial \theta_j} * \frac{1}{m}\sum_{i=1}^m \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 \quad (7)$$

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \theta_2) = \frac{\partial}{\partial \theta_j} \frac{1}{m}\sum_{i=1}^m \left(\theta_0 + \theta_1 x^{(i)} + \theta_2 x^{2(i)} - y^{(i)}\right)^2 \quad (8)$$

$$\theta_0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1, \theta_2) = \frac{1}{m}\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \quad (9)$$

$$\theta_1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1, \theta_2) = \frac{1}{m}\sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})(x^{(i)}) \quad (10)$$

$$\theta_2 : \frac{\partial}{\partial \theta_2} J(\theta_0, \theta_1, \theta_2) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})(x^{2(i)})$$

Gradient descent algorithm need to repeat until convergence. Below we show code that estimates the parameters $\theta_0, \theta_1, \theta_2$.

```
//collected data of angle
float X[30]={-84.36, -86.41, -86.72, -87.4, -88.34, -88.64, -87.52, -86.82, -84.35, -83.3, -81.41, -77.88,
-76.51, -74.67, -73.28, -69.26, -67.14, -64.51, -59.92, -57.22, -54.76, -51.06, -46.15, -42.99, -39.4, -32.27,
-27.05, -22.4, -16.65, 0};
//collected data of PWM
float Y[30]={760, 766, 772,778, 784, 790, 796, 802, 808, 814, 820, 826, 832, 838, 844, 850, 856, 862, 868,
874, 880, 886, 892, 898, 904, 910, 916, 922, 928, 934};
for(int i=0; i<1000; i++)                          //θj: = θj − α ∂/∂θj J(θ0, θ1, θ2)
    {  J0=0;
       J1=0;
       J2=0;
         for(int j=0; j<30; j++)                   //∑_{i=1}^{m}(θ0 + θ1x^(i) + θ2x^2(i) − y^(i))^2
         {
         xh =X[j]/100;                             //register get overflowed during calculation
         yh =Y[j]/100;                             //register get overflowed during calculation
         hyp = theta0+(theta1*xh)+(theta2*xh*xh);  // hθ(x) = θ0 + θ1x + θ2x^2
         J0=J0+(hyp - yh);                         //∂/∂θ0 J(θ0, θ1, θ2,)
         J1=J1+(hyp - yh)*xh;                      //∂/∂θ1 J(θ0, θ1, θ2,)
         J2=J2+(hyp-yh)*xh*xh;                     //∂/∂θ2 J(θ0, θ1, θ2,)
         }
theta_zero = theta0 - (0.02 * J0);                 //tθ0: = θ0 − α ∂/∂θ0 J(θ0, θ1, θ2,)
theta_one  = theta1 - (0.02 * J1);                 //tθ1: = θ1 − α ∂/∂θ1 J(θ0, θ1, θ2,)
theta_two  = theta2 - (0.02 * J2);                 //tθ2: = θ2 − α ∂/∂θ2 J(θ0, θ1, θ2,)
theta0=theta_zero;                                 //θ0  updated
theta1=theta_one;                                  //θ1  updated
theta2=theta_two;                                  //θ2  updated
    Jc=0;
    Jh=0;
 for(int k=0; k<30; k++);                          //J(θ0, θ1, θ2) = 1/2m ∑_i^m (hθ(x^(i)) − y^(i))^2
        {
        xh =X[k]/100;
        yh =Y[k]/100;
        hyp = theta0+(theta1*xh)+(theta2*xh*xh);
        Jc+=(hyp-yh)*(hyp-yh);
        }
        Jh=Jc/30;
```

}

5. Experimental Result

Propeller based single pendulum shown in Fig. 1 that swings between -90º - +10º. We have collected 30 training data (Table 1) and parameters are defined by computing on the microcontroller. The computation were performed on the Atmega32 controller that operates at 8MHz with learning rate α=0.02. The following parameters were calculated as result of machine learning.

$$\theta_0 := 926{,}67$$
$$\theta_1 := -0{,}4046$$
$$\theta_2 := -0{,}0229$$

Now hypothesis function is formulated as:

$$h_\theta(x) = 926.67 - 0.4046x - 0.0229x^2$$

The result of the estimated polynomial function is shown in Fig. 5 in green.

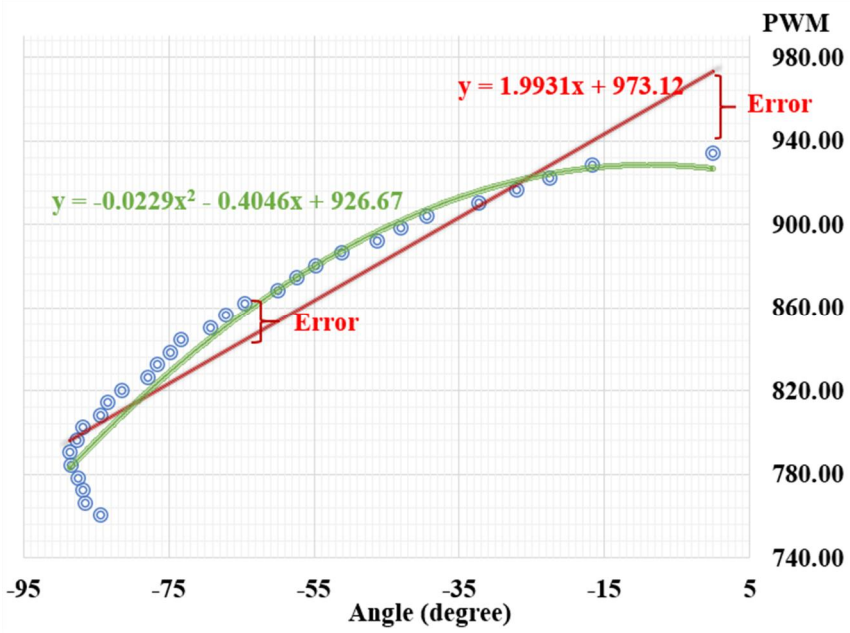

Figure 4. Estimated polynomial function

6. Conclusion

The single pendulum equipped by propeller system was trained and successfully controlled. The learning speed, α, was determined by experiments. If we choose $\alpha \geq 0.02$ divert occurs. The following two items depend on the selection of the controller.

First, the size of the learning data depends on the size of the memory. The higher the number of learning data the more likely the error will be smaller. During the calculation using floating point data register overflowed.

Second, calculation of the parameters depends on the speed at which the controller works. 600000 repeated loops for estimation of the parameters took about 90 seconds.

Controller with machine learning algorithm was able to hold pendulum in given angle. The system was able to hold in given angle by re-learning way with change of power voltage or weight of the pendulum. In the future, it is possible to add integral feedback to the system to improve the system and reduce the errors.

References

[1] https://en.wikipedia.org/wiki/Machine_learning

[2] Andrew Ng. *CS229: Machine Learning* course. Computer Science Department, Stanford University.

[3] R.S. Michalski, J.G. Carbonell, T.M. Mitchell. "*Machine Learning: An Artificial Intelligence Approach*". 1983

[4] S. B. Kotsiantis. *Supervised Machine Learning:* A Review of Classification Techniques. *Informatica 31.* 249-268, 2007

[5] https://www.coursera.org/learn/machine-learning

[6] Ayon Dey. Machine Learning Algorithms: A Review. *International Journal of Computer Science and Information Technologies*. Vol. 7 (3) , 2016, 1174-1179

[7] https://en.wikipedia.org/wiki/Supervised_learning

[8] Tengis Tserendondog, Batmunkh Amar. *"Study of a balancing system based on stereo image processing",* MUST, The compilation of science works of professors and teachers. 2015, №19/183, pages 268-274

[9] Tengis Tserendondog, Batmunkh Amar. *"PID and State Space Control of Unbalanced Swing* ". Mongolian Information Technology-2016. The compilation of Science Conference. Page 125.

[10] Tengis Tserendondog, Batmunkh Amar, Byambajav Ragchaa. "*Stereo Vision Based Balancing System Results*". International Journal of Internet, Broadcasting and Communication. Vol.8 No.1, 1-6. E-ISSN number, 2288-4939.

[11] Amar Batmunkh, Tserendondog Tengis. "*State feedback control of unbalanced seesaw*". 11th International Forum on Strategic Technology (IFOST), 2016.
DOI: https://ieeexplore.ieee.org/document/7884181/

[12] Tengis Tserendondog, Batmunkh Amar. " *Quadcopter stabilization using state feedback controller by pole placement method*". International Journal of Internet, Broadcasting and Communication Vol.9 No.1, 1-6, E-ISSN number, 2288-4939,

[13] Tengis Tserendondog, Batmunkh Amar. "*Disturbance Rejection Control for Unbalanced Double-Propeller System on Single Axis*". Khureltogoot-2017, Proceeding of International Conference of Technology and Innovation, pages 20-23. Ulaanbaatar.

[14] Jannick Verlie. *Control of an inverted pendulum with deep reinforcement learning.* Master's dissertation. Department of Electronics and Information Systems. Ghent University.

[15] Ciro Donalek. *Supervised and Unsupervised Learning*. Lecture Note. 2012.

[16] Kangbeom Cheon, Jaehoon Kim, Moussa Hamadache, Dongik Lee. *"On Replacing PID Controller with Deep Learning Controller for DC Motor System"*. Journal of Automation and Control Engineering Vol. 3, No. 6, December 2015

[17] Hristo Novatchkova, Arnold Bacaa. "*Machine learning methods for the automatic evaluation of exercises on sensor-equipped weight training machines*". 9th Conference of the International Sports Engineering Association. 2012

[18] Vijaykumar Gullapalli. " *A comparison of supervised and reinforcement learning methods on areinforcement task*". Computer and Information Science Department, University of Massachusetts.

[19] C. W. Anderson. *Learning to control an inverted pendulum using neural networks*. IEEE Control System Magazine, 9(3): 31-37, 1989.

[20] Seongsoo Cho, Bhanu Shrestha, "*A Study on Accuracy Estimation of Service Model by Cross-validation and Pattern Matching*", Quadcopter stabilization using state feedback controller by pole placement method". International Journal of Internet, Broadcasting and Communication. Vol.7 No.3, 2017.

[21] Martin Riedmiller. *Neural Reinforcement Learning to Swing-up and Balance a Real Pole*. Neuroinformatics Group University of Osnabrueck. 2000.