Revista Brasileira de Ensino de Física, vol. 45, e20230349 (2023)
www.scielo.br/rbef
DOI: https://doi.org/10.1590/1806-9126-RBEF-2023-0349

Using neural networks to predict the range of projectile motions in viscous medium

José Carlos Ferreira Bastos[1], Aderaldo Irineu Levartoski de Araujo[*1], Jhonny Leite Martins de Sousa[2], Glendo de Freitas Guimarães[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Departamento de Física e Matemática, Fortaleza, CE, Brasil.
[2]Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Departamento de Telemática, Fortaleza, CE, Brasil.

Received on November 12, 2023. Accepted on November 20, 2023.

The motion of any material body after be launched from some point in Earth surface start to follow a trajectory determined by two different interactions, the gravitational attraction and the resistance to his movement caused by the contact with the molecules of the medium in which it travels. Considering that a huge amount of physics problems remains not even mentioned on physics courses due to the requirements of strong mathematical background or computational effort. The authors propose a new and easier approach to introduce the study of complex problems in introductory physics courses, based on neural networks and machine learning. Through one simple example of classical mechanics with artificial intelligence the authors are supposed to contribute with the introduction of new technologies to physics learning.
Keywords: Projectile motion, horizontal range, neural networks, viscous medium.

1. Introduction

The use of artificial intelligence has taken on considerable proportions in our daily lives in recent years and caused transformations with relevant social and economic impact [1]. The increasing speed of information processing combined with the development of efficient data transmission and storage systems has leveraged major advances in areas such as imaging diagnostics, arts, economics, automation, pattern recognition, social network analysis and data security, among others [2]. Adaptation to this powerful technological resource has occurred quickly, which causes a need to introduce artificial intelligence in the area of physics teaching [3–5]. This demand has presented challenges to teachers and educational institutions, which naturally occurs when emerging technologies are aimed at innovation in teaching [6].

The development of appropriate approach methods and the availability of a specialized bibliography are spaces to be filled in the current stage of this knowledge area. This work seeks to present a contribution in this sense by describing a method for studying the oblique launch of an object in a viscous medium. The problem was chosen with the didactic objective of using the students' familiarity with oblique launch without air resistance, so that understanding the role of the viscous friction force proportional to the speed of the object becomes more direct and intuitive.

A database was initially generated from mathematical modeling of the addressed problem using conventional numerical computing. This data was then used to train a neural network built for the considered system. Our objective is to present a proposal for innovation in the computational approach to solving physics problems for teaching purposes. By presenting this contribution, we intend to contribute to disseminating artificial intelligence as a teaching tool in the area of physics.

2. Oblique Launch

The oblique launch problem has numerous applications from Olympic sports activities to the launch of georeferencing satellites and aerospace probes [7].

The dependence of the air damping force with velocity is quadratic for objects moving with high velocities. However, relatively small objects with velocities lower than approximately $24m/s(86km/h)$ faces air damping depending on $-bv$ [17]:

$$\sum \vec{F_y} = m\vec{a_y} \tag{1}$$

$$\sum F_y = mg - bv_y = ma_y \tag{2}$$

$$x(t) = \frac{m}{b}v_{0x}(1 - e^{\frac{-b}{m}t}) \Rightarrow \frac{bx}{mv_{0x}} = 1 - e^{\frac{-b}{m}t}$$

$$\Rightarrow e^{\frac{-b}{m}t} = 1 - \frac{bx}{mv_{0x}} \Rightarrow -\frac{b}{m}t = \ln\left(1 - \frac{bx}{mv_{0x}}\right)$$

$$\Rightarrow -\frac{m}{b}\ln\left(1 - \frac{bx}{mv_{0x}}\right) \tag{3}$$

*Correspondence email address: aderaldo@ifce.edu.br

Substituting t found in the equation y(t), we have:

$$y(t) = \left(\frac{m^2}{b^2}g + \frac{m}{b}v_{0y}\right)\left(1 - e^{\frac{-b}{m}t}\right) - \frac{m}{b}gt$$

$$\Rightarrow y(x) = \left(\frac{m^2}{b^2}g + \frac{m}{b}v_{0y}\right)$$

$$= \left\{1 - e^{\frac{-b}{m}t}\left[-\frac{m}{b}\ln\left(1 - \frac{bx}{mv_{0x}}\right)\right]\right\} +$$

$$-\frac{m}{b}g\left[-\frac{m}{b}\ln\left(1 - \frac{bx}{mv_{0x}}\right)\right] \quad (4)$$

$$\Rightarrow y(x) = \left(\frac{m^2}{b^2}g + \frac{m}{b}v_{0y}\right)\left\{1 - e^{\ln\left(1-\frac{bx}{mv_{0x}}\right)}\right\} +$$

$$+\frac{m^2}{b^2}g\ln\left(1 - \frac{bx}{mv_{0x}}\right) \quad (5)$$

$$\Rightarrow \frac{m^2}{b^2}g\frac{m}{b}v_{0y}\Big)\left[1 - 1 + \frac{bx}{mv_{0x}}\right] +$$

$$+\frac{m^2}{b^2}g\ln\left(1 - \frac{bx}{mv_{0x}}\right) \quad (6)$$

But $v_{0x} = v_0\cos\theta_0;\ v_{0y} = v_0\sin\theta_0$

$$y(x) = \left(\frac{mg}{bv_0\cos\theta_0} + \tan\theta_0\right)x$$

$$+\ \frac{m^2}{b^2}g\ln\left(1 - \frac{bx}{mv_0\cos\theta_0}\right) \quad (7)$$

Expanding function (3), the term which appears as the logarithm, we will have:

$$\ln\left(1 - \frac{bx}{m\cdot v_0\cdot\cos\theta_0}\right) \quad (8)$$

With $\omega = \frac{bx}{mv_{0x}}$, we have:

$$\ln(1-\omega) \approx\ 0 + \frac{(-1)}{1!}\omega^1 + \frac{(-1)}{2!}\omega^2 + \frac{(-2)}{3!}\omega^3\cdots \quad (9)$$

$$\ln(1-\omega) \approx\ -\frac{\omega^1}{1} - \frac{\omega^2}{2} - \frac{2\omega^3}{3\cdot 2}\cdots \quad (10)$$

$$\ln(1-\omega) \approx\ -\omega - \frac{\omega^2}{2} - \frac{\omega^3}{3}\cdots \quad (11)$$

Then $\omega = \frac{bx}{mv_{0x}}$, the expansion becomes:

$$\ln\left(1 - \frac{bx}{mv_{0x}}\right) \approx -\frac{bx}{mv_{0x}} +$$

$$-\ \frac{b^2x^2}{2m^2v_{0x}^2} - \frac{b^3x^3}{3m^3v_{0x}^3}\cdots \quad (12)$$

Substituting the expansion into the trajectory equation, we have:

$$y(x) = \left(\frac{mg}{b\cdot v_0\cdot\cos\theta_0} + \tan\theta_0\right)x$$

$$+\ \frac{m^2}{b^2}g\left(\frac{-bx}{mv_{0x}} - \frac{b^2x^2}{2m^2v_{0x}^2} - \frac{b^3x^3}{3m^3v_{0x}^3}\right)$$

$$=\tan\ \theta_0 x - \frac{g}{2v_0^2\cos^2\theta}x^2 - \frac{bg}{3mv_0^3}x^3 = 0 \quad (13)$$

3. Neural Networks

Modern computers have the ability to simulate circuits similar to networks. The basic principle of a neural network is schematically depicted in Figure 1, given by one or more input variables, hidden layers of different sizes and quantities, which can change according to the objective, and also the layer of one or more outputs. The signals emitted by neurons are represented by the formula:

$$S_k = f\left(\left(\sum_1^n x_i\cdot w_{ik}\right) + b_k\right) \quad (14)$$

S_k: signal emitted by neuron k
f: activation function
n: number of entries from the previous layer
x_i: i-th entry
w_{ik}: weight between presynaptic neuron i and postsynaptic neuron k
b_k: bias associated with neuron k

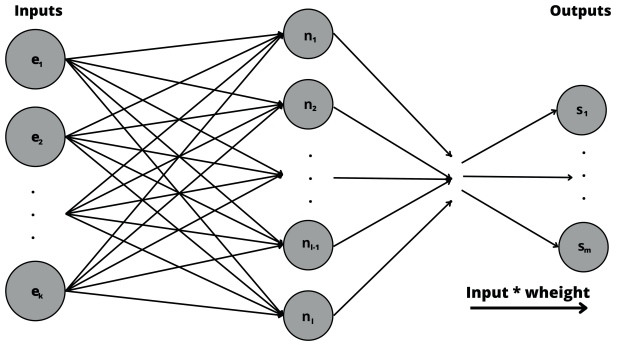

Figure 1: Basic structure of a neural network with k inputs, l neurons, and m outputs.

Random weights associated with all variables are initially used to conduct the first iterations (epochs). An epoch carried out with a batch of input values interacts to the network and return a set of predicted ranges. Such random weights lead to outputs far from the expected values. The mean squared error is applied to evaluate batch performance as follow:

$$MSE = \frac{1}{n}\sum_1^n (y_i - p_i)^2 \quad (15)$$

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2023-0349

n: number of samples in the batch
y_i: actual sample value
p_i: predicted value

Based on the mean squared error found, the RMSProp optimization algorithm [9, 10] is used to calculate the new weights that will replace the current ones, as indicated in Figure 2, so that the MSE value can be reduced in the next epochs. The formula for this algorithm is given by:

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)\left(\frac{\delta C}{\delta w}\right)^2$$
$$\rightarrow w_{t+1} = w_t - \frac{\eta}{\sqrt{E[g^2]_{t+1}}}\frac{\delta C}{\delta w} \qquad (16)$$

w_{t+1}: updated weight
w_t: current weight
η: learning rate, example = 0.001
$E[g^2]$: moving average of the quadratic gradient
β: moving average parameter
$\frac{\delta C}{\delta w}$: gradient of the cost function with weight

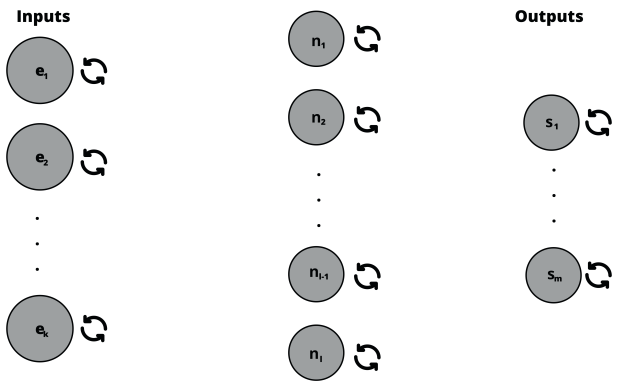

Figure 2: Update of all weights.

4. Method

Table 1 shows data from several tests carried out computationally using the Python programming language [11] and the Sympy library [12], aimed at solving equations, and all results were stored in a spreadsheet. With the input variables (mass, initial velocity, gravity, angle and damping constant), Figure 3 stands for the better range calculated and the result (x) which is the correct range calculated, we have a large table of information formed by 4 columns of inputs and 1 for output.

From the launch equation:

$$\tan\ \theta_0 x - \frac{g}{2v_0^2\cos^2\theta}x^2 - \frac{bg}{3mv_0^3}x^3 = 0 \qquad (17)$$

m: mass
v_0: initial velocity
θ: launch angle
b: damping constant

Table 1: Empirical data for neural network training.

Mass	Angle	v_0	b	Range
0.02	10	15	1	1.65699250856122
0.02	10	16	1	1.83227485990107
0.02	10	17	1	2.01356601369026
0.02	10	18	1	2.20068717297175
–	–	–	–	–
–	–	–	–	–
–	–	–	–	–
0.09	89	22	4.8	0.131135770172690
0.09	89	23	4.8	0.140299594234527
0.09	89	24	4.8	0.149670151636403
0.09	89	25	4.8	0.159243089770685

Equations were calculated with all possible combinations between values:

$$1 <= b <= 4.8 \mid \text{Step} = 0.2$$
$$10 <= \theta <= 89 \mid \text{Step} = 1$$
$$0.02 <= m <= 0.09 \mid \text{Step} = 0.01$$
$$15 <= v_0 <= 25 \mid \text{Step} = 1$$

The TensorFlow library [13], an open-source software library was applied to build the network. Such application offers high performance topological resources to implement machine learning models and neural networks. In addition, it is designed to training complex neural networks for tasks like image recognition, language processing, numerical predictions and time series analysis. It menages to handle with large data sets and intense mathematical operations. One of the main advantages of TensorFlow is its ability to handle large volumes of data and intense mathematical operations. It enables defining high-level models using abstractions such as layers and computational graphs, and provides a rich Application Programming Interface (API) for customization and extensibility.

The first step in executing training is to pre-process the data, initially separating it into training data and testing data in an 80/20 ratio. After this, we must also separate the target values (range) from the other variables to be the supervised learning labels [14].

From this point on we can perform data normalization, which consists of transforming the values into a specific scale in a standardized way which facilitates comparison with other batches of data in order to avoid the existence of learning biases and improve model convergence. The normalization formula is given by:

$$Z = \frac{x-\mu}{\sigma} \qquad (18)$$

Z: normalized value
x: value
μ: mean of the values
σ: standard deviation of the values

After normalizing the input data, we use the Keras API which runs on top of Tensorflow [15]. It was possible to create a network using the keras function. Sequential()

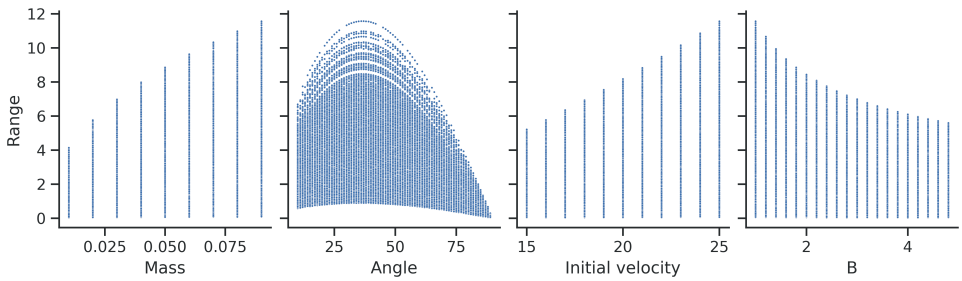

Figure 3: Range depending on the parameters in Table 1.

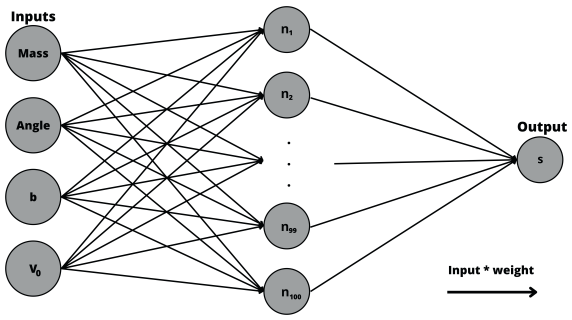

Figure 4: Structure of the neural network adapted to the problem.

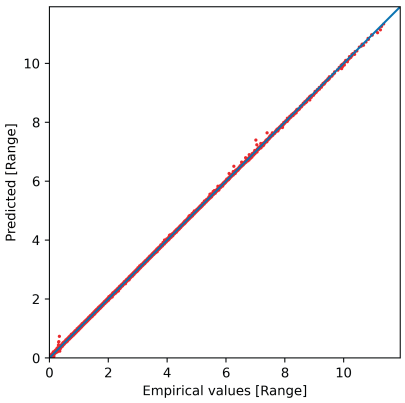

Figure 5: Comparison between the values found by the network with the correct values that were labeled.

is responsible for grouping the layers of the network of Figure 4, defining only one internal layer with 100 neurons, all connected to an output, which represents the predicted range. The activation function applied was that which returns the maximum value between 0 and the signal received by the neuron.

With the network parameters defined, we then use the fit method added with normalized training data and labels with the results of their combinations to perform training for 1000 epochs, recalculating the weights through backpropagation in each one.

5. Results

We created a neural network which learned from the data that was delivered, as shown in Figure 5, and was then able to predict the results of new launches with values very close to those obtained when solving equations, which can be observed in Table 2.

We can evaluate the performance of a neural network with the root mean squared error evaluation metric (RMSE) [16], which is very useful to check its prediction quality. The lower the result, the better the network's ability to estimate values. It is widely used in machine learning implementations, and its formula is:

The square root of the last MSE value obtained equals the RMSE

$$\sqrt{0.00048722} \approx 0.02207316$$

Such a small magnitude of the RMSE represents how appropriate neural networks can be to make the study of mathematically sophisticated physics problems much more easier to understand in learning processes.

Table 2: Random empirical examples with their predictions.

Mass	Angle	v_0	b	Range	Predicted
0.05	87	23	1.2	0.615001	0.679726
0.09	68	19	3	2.707170	2.725363
0.01	57	19	1.4	1.876964	1.886901
0.09	57	19	3.6	3.424880	3.418013
0.04	29	23	2.8	4.231864	4.242589
0.08	18	24	3.6	4.760582	4.760784

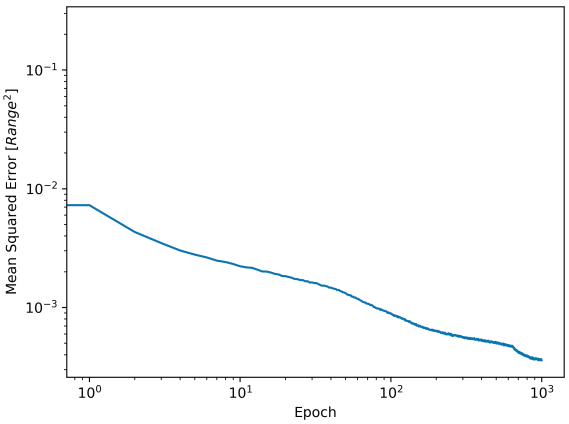

Figure 6: Evolution of MSE during epochs.

 DOI: https://doi.org/10.1590/1806-9126-RBEF-2023-0349

6. Final Considerations

Considering the revolutionary paradigm shift now in course, carried out by different ways of perform machine learning and apply it to different branches of human activity, allied to author's intent of popularize artificial intelligence in science education, the authors performed the machine learning approach in a very simple classical mechanics problem, and of course the results obtained might be considered very good.

References

[1] P. Aghion, C. Antonin and S. Bunel, Economie et Statistique/Economics and Statistics **510-511-512**, 150 (2019).

[2] S. Das, A. Dey, A. Pal and N. Roy, International Journal of Computer Applications **115**, 31 (2015).

[3] O. Zawacki-Richter, V.I. Marín, M. Bond and F. Gouverneur, International Journal of Educational Technology in Higher Education **16**, 1 (2019).

[4] G.R. Schleder and A. Fazzio, Revista Brasileira de Ensino de Física **43**, e20200407 (2021).

[5] H. Ferreira, E.F. Almeida Junior, W. Espinosa-García, E. Novais, J.N.B. Rodrigues and G.M. Dalpian, Revista Brasileira de Ensino de Física **44**, e20220214 (2022).

[6] Z.A. Krusberg, Journal of Science Education and Technology **16**, 401 (2007).

[7] W. Freire, M. Medeiros, D. Leite and R. Silva, Revista Brasileira De Ensino De Física **38**, 1306 (2016).

[8] M. Sousa, *Como funcionam os neurônios?*, available in: https://www.colegioweb.com.br/biologia/como-funcionam-os-neuronios.html

[9] V. Bushaev, *Undestanding RMSprop - faster neural networking learning*, available in: https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a

[10] T. Tieleman and G. Hinton, COURSERA: Neural Networks for Machine Learning **4**, 26 (2012)

[11] G. Van Rossum and F. Drake Jr, *Python reference manual* (Centrum voor Wiskunde en Informatica, Amsterdam, 1995).

[12] A. Meurer, C. Smith, M. Paprocki, O. Čertík, S. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. Moore, S. Singh et al. *PeerJ Computer Science* **3**, e103 (2017).

[13] M. Abadi, P. Barham, J. Chen, Z. Chen and A. Davis, in: *12th USENIX Symposium On Operating Systems Design And Implementation (OSDI 16)* (Savannah, 2016).

[14] T. Ludermir, Estudos Avançados **35**, 85 (2021).

[15] F. Chollet, Q.S. Zhu, F. Rahman, T. Lee, C. Qhian, G. Marmiesse, H. Jin, O. Zabluda, S. Marks and M. Watson et al. *Keras, GitHub, (2015)*, available in: https://github.com/fchollet/keras

[16] D. Santos and E. Oliveira, Sociedade Brasileira De Automática **1**, SBSE2020 (2020).

[17] S.T. Thornton and J.B. Marion, *Classical dynamics of particles and systems* (Boston, Cengage, 2020).