An Incursion to Deep-Learning for Process Regulation

Una incursión al aprendizaje profundo para la regulación de procesos

A. Mosso-Vazquez, A. Hernandez, G. Nagarajan, D.Juarez-Romero

[1] Departamento de Mecatrónica, Instituto Tecnológico Superior de Calkiní en el Estado de Campeche Av. Ah-Canul S/N Col. San Felipe CP 24900, Calkiní, Campeche, México.

[2] Centro de Investigación en Ingeniería y Ciencias Aplicadas,UAEM Av. Universidad No. 1001, Col. Chamilpa, Cuernavaca, Morelos, CP 62209, México.

[3] I.C.E. Division, Department of Mechanical Engineering, Anna University,Chennai 600 025 India.

* Correo-e: amosso@itescam.edu.mx

KEYWORDS:

Model construction, Model Validation, Predictive Model, Deep Learning, Reinforcement Learning.

ABSTRACT

A model which represents a physical process is usually composed by conservation equations, transfer mechanisms, and closure equations. These equations vary in the degree of certainty. This paper describes the incorporation of physical and empirical models. The empirical part is constructed by Deep Learning. This work describes the principles which have promoted Deep Learning as a complementary tool for the approximation of process engineering when is used for model-based control. In addition of the stability and accuracy to deal with unmeasured disturbances, a robust strategy is to use Reinforcement Learning thus the principles of this strategy are also described.

PALABRAS CLAVE:

Construcción de modelos, Validación de modelos, Modelo Predictivo, Aprendizaje Profundo, Aprendizaje por Refuerzo.

RESUMEN

Un modelo que representa un proceso físico suele estar compuesto por ecuaciones de conservación, mecanismos de transferencia y ecuaciones cerradas. Estas ecuaciones varían en el grado de certeza. Este artículo describe la incorporación de modelos físicos y empíricos. La parte empírica está construida por Aprendizaje Profundo. Este trabajo describe los principios que han impulsado al Aprendizaje Profundo como herramienta complementaria para la aproximación de la ingeniería de procesos cuando se utiliza para el control basado en modelos. Además de la estabilidad y precisión para hacer frente a perturbaciones no medidas, una estrategia robusta es utilizar el Aprendizaje por Refuerzo. Por lo tanto, también se describen los principios de esta estrategia.
.

Recibido: 29 de diciembre de 2020 • **Aceptado:** 13 de mayo de 2021 • **Publicado en línea:** 4 de junio de 2021

1 Introduction

In this paper an incursion to deep learning and reinforcement learning for processes regulation is presented. In section 2 some nonlinear objective functions used in deep neural network are introduced. In section 3 the gradient method is presented as a solution method of the objective function. A test of deep learning approximation is presented in section 4. In section 5, applications to heating processes is presented, and conclusions are presented in section 6. In this section, we first review the main ideas that lead to deep learning by means of a brief history; then, we present the basic definitions and concepts of reinforcement learning with the aim to only state the fundamentals for processes regulation.

1.1 A brief review of the history of deep learning

There have been many contributions to the field of neural network that lead to what we know as deep learning algorithms. However, we present next the most remarkable ideas, which start with the work of Mcculloch and Pitts in 1943 and stop with the work of Hinton et al in 2006 [1 – 7].

W. MCCULLOCH AND W. PITTS [1] in1943 introduced the first mathematical model of a neuron, in which a weighted sum of input signals is compared to a threshold to determine whether or not the neuron fires. It should be noted that the weight in this model were adjusted manually.

F. ROSENBLATT [2, 6] in 1958 introduces the perceptron. The perceptron occupies a special place in the historical development of neural networks because of the stated hypothesis on the perceptron and because it was the first neural network to be described algorithmically. That is, this model could automatically learn the weights needed to classify an input, without human intervention.

M. MINSKY AND S. PAPERT [3, 6] in 1969 showed the first rigorous study dedicated to determining what a perceptron network is capable of learning. They predicted perceptron limitations, which diminish the research on neural networks.

D. E. RUMELHART, G. E. HINTON, and R. J. WILLIAMS [4, 6] in 1986 presented a key influence in the resurgence of interest in the field of neural networks. They introduced the backpropagation algorithm for multilayer network training. That is, the back propagation of the error enabled us to add hidden layers obtaining, therefore, deep neural networks. The difficulty faced at this time was the lack of algorithms to solve the multilayer networks efficiently.

G. E. HINTON, S. OSINDERO AND Y-W THE [5] in 2006 published a paper showing how to train efficiently a deep neural network. They called this technique "Deep Learning". They revived the interest of the scientific community. New works proved that deep learning was not only possible, but capable of spectacular achievements that no other Machine Learning technique could obtain. The key properties of deep learning methodology are its efficient algorithms, which have been enhanced by the current computational capacities. HINTON was contributing to the achievements of deep neural network around 1986 as it may be noticed in [4].

1.2 A review of reinforcement learning

The reinforcement learning covers Markov process (MP), Markov process with rewards (MPR), Markov decision processes (MDP) and dynamic programming. Reinforcement learning have been widely used in modeling disciplines such as economics, control theory, and robotics. We first present the basic definitions and concepts of reinforcement learning, including the agent, environment, action, state, and the reward function. Then, we review the MDP together with MP and MRP because they are the cornerstones in formulating reinforcement learning tasks [7].

1.2.1 Definitions and concepts of reinforcement learning

The agent and environment are the basic components of reinforcement learning. An agent can "interact" with the environment by using a predefined action set $A = \{A_1, A_2, \cdots\}$. The goal of reinforcement learning algorithms is to teach the agent how to interact with the environment, as it is shown in Figure 1.

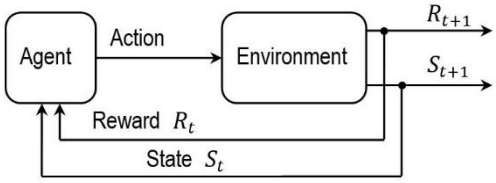

Fig. 1 Agent and environment

At any time step t the agent first observes the current state of the environment S_t and the corresponding reward value R_t. Then it decides what action to take next based on S_t and R_t. The action A_t the agent perform is fed into the environment giving the new state S_{t+1} and reward R_{t+1}. If the observation of the environment only contains partial state information, the environment is partially observable, otherwise is fully observable. A trajectory τ is a sequence of states, actions, and rewards:

$$\tau = (S_0, A_0, R_0, S_0, A_0, R_0, \cdots)$$

which records how the agent interacts with the environment. An episode is a trajectory that goes from an initial state to the terminal state.

The transition from a state to the next state can be either deterministic or stochastic. For a deterministic transition, the next state S_{t+1} is governed by a deterministic function: $S_{t+1} = f(S_t, A_t)$. For a stochastic transition, the next state S_{t+1} is described as a probabilistic distribution:

$$S_{t+1} = p(S_{t+1}|S_t, A_t)$$

1.2.2 Fundamentals of reinforcement learning

Markov Process

A Markov process (MP) is a discrete stochastic process, which follows *the assumption of Markov chain* where the next state S_{t+1} is only dependent on the current state S_t of an environment, with the transition probability is described as follows:

$$p(S_{t+1}|S_t) = p(S_{t+1}|S_0, S_1, S_2, \cdots, S_t) \quad (1)$$

A Markov chain is *time-homogeneous Markov chain* if it holds the following property $p(S_{t+2} = s'|S_{t+1} = s) = p(S_{t+1} = s'|S_t = s)$.

Markov Reward Process

An agent can interact with its environment via the state transition matrix $P = (p_{ij})$. There is no way, however for MP to provide feedback from the

environment to the agent. To provide a feedback, Markov reward process (MRP) extends MP from (S, P) to (S, P, R, γ). The R and γ represent the reward function and reward discount factor, respectively. The reward function depends on the current state:

$$R_t = R(S_t) \tag{2}$$

A return is the cumulative reward of a trajectory τ, which is defined as follows:

$$R(\tau) = \sum_{t=0}^{T} R_t \tag{3}$$

The discounted return is a weighted sum of rewards of a $T-$ step trajectory for MRP, which is defined as follows:

$$R(\tau) = \sum_{t=0}^{T} \tau^t R_t \tag{4}$$

where a reward discount factor $\gamma \epsilon [0, 1]$.

Markov Decision Process

MP can be defined as the tuple (S, P), where the element of state transition matrix P is $p(S_{t+1} = s'|S_t = s)$, and MRP is defined as the tuple (S, P, R, γ). Here, MDP is defined as the tuple (S, A, P, R, γ). An element of state *transition matrix P* becomes:

$$\begin{aligned} p(s'|s, a) = p(S_{t+1} = s'|S_t \\ = s, A_t \\ = a) \end{aligned} \tag{5}$$

The set of actions is $A = \{a_1, a_2, \dots\}$. The *immediate reward* becomes:

$$R_t = R(S_t, A_t) \tag{6}$$

A policy $\pi(a|s)$ represents the way in which the agent behaves based on its observations of the state the environment:

$$\begin{aligned} &\pi(a|s) \\ &= \pi(A_t = a|S_t = s), \forall t \end{aligned} \tag{7}$$

The probability of a $T-$ step trajectory for MDP is determined in terms of the initial state probability ρ_0 and the policy π as follows:

$$\begin{aligned} &p(\tau|\pi) \\ &= \rho_0(S_0) \prod_{t=0}^{T-1} p(S_{t+1}|S_t, A_t)\, \pi(A_t|S_t) \end{aligned} \tag{8}$$

Now, given the reward function R and all possible trajectories τ, *the expected return* $J(\pi)$ is defined in terms of $p(\tau|\pi)$ as follows:

$$J(\pi) = \sum_{\tau} p(\tau|\pi) R(\tau) \tag{9}$$

The Reinforcement Learning optimization problem is to improve the policy π for maximizing $J(\pi)$. The optimal policy π^* is expressed as:

$$\pi^* = arg \max_{\pi} J(\pi) \tag{10}$$

Given π, *the value function* $V(s)$, the expected return under the state, can be defined as:

$$\begin{aligned} &V^{\pi}(s) = E_{\tau \sim \pi}(R(\tau)|S_0 = s) \\ &= E_{A_t \sim \pi(\cdot|S_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t)|S_0 = s \right] \end{aligned} \tag{11}$$

where $\tau \sim \pi$ means the trajectories τ are sampled given the policy π, $A_t \sim \pi(\cdot|S_t)$ means the action under a state is sampled from the policy. *The action-value function* gives an expected return under a state and an action. If the agent acts according to a policy π, we denote it as $Q^{\pi}(s, a)$, which is defined as:

$$Q^{\pi}(s,a) = E_{\tau\sim\pi}(R(\tau)|S_0 = s, A_0 = a) = E_{A_t\sim\pi(\cdot|S_t)}\left[\sum_{t=0}^{\infty}\gamma^t R(S_t,A_t)|S_0 = s, A_0 = a\right] \quad (12)$$

The estimation formulas of value functions $V^{\pi}(s)$ or $Q^{\pi}(s,a)$ can be simplified by leveraging the Markov property in MDP, which may be solved by dynamic programming.

1.3 Dynamic programming

Dynamic programming (DP) provides a general framework for a dynamic problem by breaking it down into sub-problems. There are two properties that a problem must have for DP to be applicable: optimal substructure and overlapping sub-problems. A MDP with finite actions and states satisfy both properties. *The Bellman equation* gives the recursive decomposition of a MDP, and value functions provides the optimal solution of sub-problems which can be reused. DP requires full knowledge of the environment, such as the reward model and the transition model, of which we often have limited knowledge in reinforcement learning. However, DP provides a framework for a reinforcement learning algorithm in such a manner that it may learn to interact with the MDP incrementally.

2 Objective functions of Deep-Learning

Bengio 2000 [8] defined as objectives of Deep-Learning:

1. *General* – To treat applications with variations much greater than the number of training examples.
2. *Flexible Scope* – Ability to learn at low-level, intermediate and high level.
3. *Robust* – To learn from large set of examples.
4. *Reusable* – To learn across different tasks.
5. *Automatic* – To learn the structure of the observed data.

In the next sections, we shal discuss the progress toward these objectives.

2.1 Approximation

A linear polynomial approximation, with respect to the weight w, can be expressed as:

$$y(x_i) = \sum_{j=1}^{m} w_j \varphi_j(x_i)$$

where $\varphi_j(x_i)$ represent basis, form or shape function, and w_j are the weights.

A common basis function used by Artificial Neural Networks (ANN) is $tanh(x)$,

$$\varphi_j(x_i) = tanh(x) = \frac{e^x - e^x}{e^x + e^x}, \quad -1 \le \varphi_j(x_i) \le +1$$

This basis function has the following properties

1. Stable for the x_i values in the range $x_{min} \le x_i \le x_{max}$
2. Good convergence as $m \to \infty$.
3. Revertible $x = \varphi^{-1}(x)$. During this process it is required that if there are multiple roots, then it should exist a criterium to select the appropriate root.
4. Continuous in the derivatives $\partial\varphi/\partial x$.

For efficient calculation, ANN uses an equivalent shape, but with a shifted value. The following combined form of weights w and bias b (cf. Goodfellow, 2016 [9]):

$$\varphi(x,w) = \frac{1}{1+e^{wx+b}}, \quad 0 \le \varphi(x,w) \le 1$$

2.2 Promotion of Deep-Learning Approximation

Analysis of process phenomena presents a strong progress due to the availability of

1. Large and accurate set of data.
2. Inter-phases to interconnect different sensing devices.
3. Computers with multiprocessing capability.
4. Standard formats to exchange networks, ONNX.ai (Open Neural Network Exchange).

As a result many phenomena can be approximated by: amplitude, frequency, phase, and synchronization with other phenomena.

2.3 Approximation of Several variables

To approximate several variables ANN uses the form:

$$y_1 = \varphi_1(x^T w_1 + b_1) \tag{13}$$

$$y_2 = \varphi_2(x^T w_2 + b_2) \tag{14}$$

It is convenient to use scaled variables, in both dependent and independent variables:

$$\hat{x} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

2.4 The objective function for approximation

The objective function measures distance between the measurements and the approximation. For the Eucledian norm:

$$J_j(w,x) = \left[y_{j,Apx} - y_{j,Meas}\right]^T \left[y_{j,Apx} - y_{j,Meas}\right] \tag{15}$$

For a general application, the optimization problem is [10]:

$$\min_w J_j\left((x^j, w), y^i\right) \tag{16}$$

2.4.1 Scaling

In general the value of the objective function has terms with different magnitude. Alternatives to scale this value are:

1. To apply the uncertainty of the measurements as a weighting value.
2. To apply the norm proposed by Shannon [11]. $xlog(x)$ provides general scaling properties

2.5 A Layer Deeper

From the ANN basis

$$\varphi(x) = \frac{1}{1 + e^{wx+b}} \tag{17}$$

A second layer can be constructed as:

$$\frac{1}{1 + e^{w_0\left(\frac{1}{1+e^{w_1 x + w_1}}\right) + b_0}}$$

The advantage is that with this additional layer, the approximation error is smoother than the original function.

3 Solution Methods of the Objective function

3.1 General Algorithm

A general algorithm for the solution of the nonlinear problem is the gradient method.

1. Initialize weights $w = w^0$

for $I_t = 1 : I_{tMAX}$

2. Evaluate gradient of the objective function with respect of the weights, $\nabla J(w)$.

3. Tune α, the step length, to promote descend direction $J(w^{I_{t+1}}) < J(w^{I_t})$ [12].

4. Update weights $w^{I_{t+1}} = w^{I_t} + \alpha \nabla J(w^{I_t})$.

5. ($J < \tau$) stop

end

6. Return fitting parameters w.

3.2 Gradient Evaluation Method

The steepest descendant method is simple and reliable. But requires the partial derivatives. There are some possibilities to evaluate the gradient $\nabla J(w)$.

$$\nabla J(w) = \begin{Bmatrix} Analitical \\ Numerical \end{Bmatrix}$$

for a nonlinear approximation, the optimum values for fitting parameters θ are evaluated $dJ/d\theta = 0$:

$$\frac{dJ}{d\theta} = f(x_i, \theta)\frac{f(x_i, \theta)}{d\theta} \qquad (18)$$

3.2.1 Example

Consider a function $f(\cdot)$ of heat transfer of a separation of mixture into liquid and vapor of independent variables $A, w_1, T_1, w_2, T_2, T_w, T_a, C_3$

$$f(A, w_1, T_1, w_2, T_2, T_w, T_a, C_3) = w_1^i \cdot h_1^i - w_2^0 \cdot h_2^0 - w_3^v \cdot h_3^v - Q_{Trn} - Q_{Lss} \qquad (19)$$

where

$$Q_{Trn} = AU_{Trn}(T_{avg} - T_w)$$

$$Q_{Lss} = AU_{Lss}(T_2 - T_a)$$

$$T_{avg} = \frac{1}{2}(T_1 + T_2)$$

Q_{Trn} Heat transfer

Q_{Lss} Energy losses to environment

U Heat transfer coefficient

$h_i = h(T_i)$

Possible fitting parameters $\theta = [U_{Trn}, U_{Lss}, A_{Lss}]$

Gradients:

$$\frac{\partial f}{\partial U_{Trn}} = \frac{\partial Q}{\partial U_{Trn}} \qquad (20)$$

$$\frac{\partial f}{\partial U_{Lss}} = \frac{\partial Q}{\partial U_{Lss}} \qquad (21)$$

$$\frac{\partial f}{\partial A_{Lss}} = \frac{\partial Q}{\partial A_{Lss}} \qquad (22)$$

Also

$$\frac{\partial Q_{Trn}}{\partial U_{Trn}} = A(T_{avg} - T_w) \qquad (23)$$

$$\frac{\partial Q_{Lss}}{\partial U_{Lss}} = A(T_2 - T_a) \qquad (24)$$

$$\frac{\partial Q_{Lss}}{\partial A_{Lss}} = U_{Lss}(T_2 - T_a) \qquad (25)$$

Observations:

- Equation (21) is based on conservation of energy, formulated around 1730.
- This equation neglects the effects of pressure and volume.
- The estimation of $h(T, C)$ requires calorimetric data of the mixture. Thus, this needs mixture data, while $h(T)$ or a pure component can be accurately estimated.
- In the laboratory Q_{Lss} can be reduced. At

industrial level this is an important component.

- In general U_{Trn} has a dependence on a) flow, b) fluid of every stream and c) form of the vessel, so this is a very specific correlation for the unit.
- The effect of unmeasured variables, like humidity, which affects Q_{Lss} can be represented by a probability distribution [13].
- The parameters U_{Lss}, A_{Lss} might be difficult to evaluate separately.
- The parameter U_{Lss} depends also on environmental conditions, like humidity.

Hernandez et al. (2009) [14] compared the approximation of a Neural Network Model with a model based in physical principles applied to a Thermodynamic cycle. They observed that the ANN are very appropriate to match the experimental results, even under heat loses and pressure drops in pipes. but it has a relatively narrow operation range.

3.3 Numerical Derivative

The numerical gradient can be approximated numerically as

$$\frac{\partial J}{\partial x} = \frac{[J(x+\delta x) - J(x-\delta x)]}{2\delta x} \qquad (26)$$

δx is a tuning parameter. Bug δx promotes large approximation errors, but to small δx leads to computer rounding errors.

3.4 Automatic Differentiation

To evaluate the partial derivatives of a general expression, automatic differentiation follows the trajectory of execution from the input values, terms, subexpresion, branches to arrive to the final output values. see Grienwank;1991 [15]. The derivatives can be obtained by the algebra of differentiation:

$$f(x): \frac{df}{dx} \qquad (27)$$

$$af_1 + bf_2: a\frac{\partial f_1}{\partial x} + b\frac{\partial f_2}{\partial x} \qquad (28)$$

$$f_1 \cdot f_2: f_2\frac{\partial f_1}{\partial x} + f_1\frac{\partial f_2}{\partial x} \qquad (29$$

$$f\big(g(x)\big): \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} \qquad (30)$$

$$\tanh(x): \sec^1(x) \qquad (31)$$

$$\max(0, f(n)): \begin{cases} \frac{df}{dx} & if\ f(x) > 0 \\ 0 & otherwise \end{cases} \qquad (32)$$

The advantage of evaluating subexpressions with the computational graph, is that the evaluation follows the branches, cycles and also procedures of the computations. The computational graph can be used in two senses:

- *Forward-mode* To obtain derived properties, like occurrence matrix [16], or partial derivatives.
- *Reverse-mode* To approximate the values of inputs which produce a required output.

This is provided directly by languages which provide operator overloading . The Julia programming language provides Automatic Differentiation by the package Zygote (cf Innes; 2019) [17].

3.5 Hessian Evaluation Method

To improve convergence Method Martens (2010) [18], used the Hessian. Hessian can be evaluated also numerically, analytically, or by geometrical approximation, known as derivative approximation [19].

4 Testing

The development of the multistage switching network BBN for share memory computers provides a general scheme to interconnect the input variables with the output variables, in such a way that by switching, the relevant variables can be detected [20] (Fig 2.)[1].

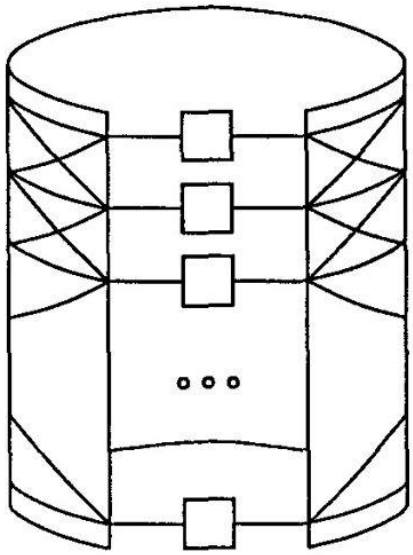

Figure 2: BBN interconnecting Network

Then with a systematic selection of variables it is possible to detect which independent variables affect more the objective function [21].

$$S_{ij} = \frac{\partial J_i}{\partial x_j} \tag{33}$$

4.1 Validation

What is the adequate level of optimization?

To improve the quality of the solution, it is possible: a) To filter the data points (see Niesser; 2020) [22], o b) To apply additional criteria, like:

1. Physical bounds of the parameters $w_{min} \leq w \leq w_{max}$
2. Sensitivities $\frac{\partial y_{Apx,i}}{\partial x_k}$ vs $\frac{\partial y_{Exp,i}}{\partial x_k}$
3. Value of Equilibrium conditions $S^*_{i,j} = S^*_{i,l}$ at $x = x^*_l$ (cf. Yuan and Herold; 2005) [23]
4. Values of Derivative at a given condition $\frac{\partial y}{\partial x}$ at $x = x^{\#}$

4.2 Overfitting

The Weierstrass approximation theorem indicates that for a given sequence of experimental values, it is possible to construct an approximation, such that $\|\partial y_{Apx} - \partial y_{Exp}\| \leq \epsilon$, but the limit might not be defined by ϵ but by the uncertainty of the measuring instruments. In this case a limit in the complexity of the approximation is necessary.

4.3 Regularization

In some experiments, exists several conditions at which the objective function has similar values. In this case a secondary criteria can be selected the optimum values of w. Then for a quadratic objective function

[1] Authors are grateful for the provision of this drawing by Prof. L. Scott

$$\min_{w}\left[J(w,x)+\sum_{l=1}^{l=NW} R_l D\left(w - w_{l,Ref}\right)^2\right] \quad (34)$$

Where $R_l > 0$ is a penalty parameter for regularization, D is a diagonal matrix, and $w_{l,Ref}$ is a reference value of w_j.

5 Applications to heating processes

The versatility of the approximation provides application in wide areas [24]

1. *Text analysis*
2. *Image processing*
3. *Sound Processing*
4. *Pattern Recognition* [26]

Also some practical applications which combine these capabilities.

1. Text analysis + sound processing to reduce noise in a conversation.
2. Image processing + sound processing to provide automobile maintenance.

Other important area is the estimation of thermodynamic properties (cf. Zu and Müller; 2020 [25], Yuan and Herold [23] for an approximation of a single potential which is used to evaluate consistently other properties), which can take more than 50% of the evaluation of a chemical process.

5.1 Process Control

Consider the dynamic form of, to represent the heating process:

$$\frac{dh}{dt} = \frac{f(A, w_1, T_1, w_2, T_2, T_w, T_a, C_3)}{M} = \frac{w_1^i \cdot h_1^i - w_2^0 \cdot h_2^0 - w_3^v \cdot h_3^v - Q_{Trn} - Q_{Lss}}{M}$$

M is the mass holdup.

If the model is accurate, then, by integration during a fixed time interval Δt, it is possible that for a given profile of inputs during time, then it is possible to predict the profile of measured variables $y_{t_0}, \cdots, y_{t_m}$.

$$y_{t_{k+1}} = \Phi(t_{k+1}, t_k) y(t_k) + \Gamma(t_{k+1}, t_k) u(t_k)$$

where:

k Sampling index

u Manipulated variable

v Unmeasured disturbances

y Measured variable

θ Model parameters

Since, a suitable control consist in following a specified reference, r_t see Table 1,

Time	**Manipulated variable**	**Disturbances**	**Measured**	**Reference**
t_0	u_0	$v(t_0)$	y_0	r_0
t_1	u_1	$v(t_1)$	y_1	r_1
t_2	u_2	$v(t_2)$	y_2	r_2
⋮	⋮	⋮	⋮	⋮

Table 1: Tabular performance of process variables over time

a suitable control strategy must reduce accumulated error over a specified predicting horizon, m, with respect the manipulated variables u_k

$$\min_{u_k \cdots u_m}\left[\sum_{i=1}^{i=m}(y_k - r_k)^T (y_k - r_k)\right] \quad (35)$$

With solution of this least square problem [27] is evaluated the manipulated variables for a predicting horizon m . The ability of agile construction of a model is quite convenient in Model Based Predictive control.

The industrial intelligence must be (cf. Qin, IFAC2020 [28]): a) Work with known first principles. To satisfy the basic continuation equations. b) Predictive: Some steps ahead. b) Interpretable to describe the rationale for the expected outcome.

A complementary part of model construction for control, is to apply reinforcement learning (Tran et al.?? 2019 [29]) and discussed in section 1, which includes:

1. Agent,
2. Environment,
3. States of the environment,
4. Transition Function (Deep-Learning).
5. Actions of the agentt on the environment,
6. Reward provided by the environment

The goal of reinforcement learning is to maximize rewards. The transition from the previous state to the next state depends on actions from the agent.

6 Conclusions

Deep-Learning offers a hierarchical representation of multivariable phenomena. Deep-Learning ensembles a set of general algorithms for approximation. The applied methodology is to classify, to select the reliable data with a recurrent stable basis function. Automatic Differentiation allows approximation with two or more layers.

6.1 Environments

Some alternatives environments are (in alphabetical order):

1. DeepLearning toolbox, by Mathworks [24].
2. *Flux*. Julia programming language, ?.
3. *Tensorflow* by Google.

6.2 Tutorials

• Goodfellow, Bengio and A. Courville (2016) [9];

• Mathwoks Academy. Deep Learning.

• Online tutorial of TensorFlow

• Online tutorial of Deep-Learning a M.I.T

• Hagan, Demuth, Beale, DeJesus: hagan.okstate.edu/nnd.html.

6.3 Nomenclature

ANN = Artificial Neural Network.

BBN = R. Bolt, L. Beranek, R. Newman switching-array.

Latins:

D Diagonal matrix

$g()$ Neural Network function

J Objective function

x Independent variable.

w Weights of the approximation

R Regularization parameter and rewards

Greeks:

φ Basis or form function

τ Tolerance value and a trajectory in MDPs

α Step length value

Subindex:

Apx Approximation

Exp Experimental

i Index of function

j Index of independent variable.

Ref Reference value

S Sensitivity and state in MDPs

min Minimum

max Maximum

Superindex:

$*$ Equilibrium condition

$\#$ Specific values

It Iteration index

This project was partially supported by colaboration project UAEM-Anna University Proy Conacyt 266752-DST2015. The authors are grateful with the organizers of the USA-Mex workshop in Numerical Analysis and Optimization (2019) Oaxaca, Mexico.

Acknowledgements

References

[1] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics.*, Vol. 5, pp. 115–133, 1943.

[2] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review*, Vol. 65, pp. 386–408, 1958.

[3] M. Minsky and S. Papert. *Perceptrons*, Cambridge, MA: MIT Press, 1969.

[4] Rumelhart, D.E., Hinton, G. E. and Williams, R. J. Learning Internal Representation by Error Propagation. In D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Cambridge, MA: MIT Press, 1986.

[5] Hinton, G. E., Osindero, S. and Yee-Whye The. A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation* 18, 1527–1554 (2006). Massachusetts Institute of Technology.

[6] Hagan, M. T., H. B. Demuth, M. H. Beale and Orlando De Jesús. *Neural Network Design*. Editorial : Martin Hagan; 2nd edition, 2014.

[7] Dong, H., Z. Ding and S. Zhang. *Deep Reinforcement Learning – Fundamentals, Research and Applications*. Springer, 2020.

[8] Bengio, Y. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, Vol. 2, No. 1 (2009) 1–127.

[9] Goodfellow I., Y. Bengio and A. Courville. *Deep Learning*. MIT Press, (2016)

[10] Amini A. *"Introduction to Deep Learning"*. MIT 6.S191 (2020)

[11] Tribus, M., & McIrvine, E. C. . Energy and information. *Scientific American*, (1971) 225(3), 179-190.

[12] Dennis, J., R.B. Schnabel R. *Numerical Methods for Unconstrained optimization and Nonlinear equations*. SIAM, (1996).

[13] Koller D., N. Friedman. *Probabilistic Graphical Models*. The MIT Press, (2009).

[14] Hernandez J.A. , R.J. Romero, D. Juárez-Romero, R.F. Escobar, J. Siqueiros. A. Neural network approach and thermodynamic model of waste energy recovery in a heat transformer in a water purification processes. *Desalination* (ISSN: 0011-9164) (2009) 243, pp 273-285

[15] Griewank A., G. F. *Corliss Automatic Differentiation of Algorithms*, SIAM(1991)

[16] Juarez-Romero D., Molina-Espinoza J. M. , Zamora-Moctezuma J. R., Leder R. Evaluating Algorithmic Properties of Dynamic Simulation Models by Operating Overloading. *Memorias de Simposium de Software y Optimización*, Cuernavaca, Mor, CICOS09, Cuernavaca (2009)ISBN 978-607-00-1970-8

[17] Innes M. , Differentiable Programming with Julia, Mar21, *London Users Group Meeting*, (2019).

[18] Martens, J. . Deep learning via hessian-free optimization. *In International Conference on Machine Learning* (ICML-10),(2010, June) Vol. 27, pp. 735-742.

[19] Nocedal, J. and Wright S. J. *Numerical Optimization*. Springer (1999)

[20] LeBlanc T.J., M. Scott and C.M. Brown. Large-Scale parallel Programming Experience with the BBN Buterfly Parallel processor. *Comp. Sci. Report, U. Rochester*, (1988).

[21] Müller, D., Esche, E., and Wozny, G. An algorithm for the identification and estimation of relevant parameters for optimization under uncertainty. *Computers & Ehemical Engineering*, (2014) 71, 94-103.

[22] Niesser M. http://niesser.github.10/12DL (2020)

[23] Yuan Z. and K.E. Herold Using a Multiproperty Free Energy Correlation. *HVAC&R Research*, (2005) v 11, 3 p377-393.

[24] Beale M.H., M.T. Hagan, Howard B. Demuth. *Deep Learning Toolbox*. UG, Mathworks (2020).

[25] Zu K. , E. A. Müller. Generating a Machine Learned Equation of State for Fluid Properties. *ArXiv*, (2020).

[26] Bishop C. *Pattern Recognition and Machine learning*. Springer.(2006)

[27] Björk A., *Numerical Methods for LeastSquares Problems*. SIAM (2006)

[28] Qin S.J. Integrated Framework of Systems, Data, and Industrial Intelligence towards Industry 4.0. *Abstract of Plenary talk - IFAC conference*, july, Germany (2020).

[29] Tran, T., Marsh, L., & Hunjet, R. Reinforcement Learning with Model Predictive Control-Recent Development. *Conference paper, ICOCTA*, Sidney (2019).

Acerca de los autores

Alejo Mosso Vázquez recibió el grado de Ingeniero en Comunicaciones y Electrónica de la Escuela Superior de Ingeniería Mecánica y Eléctrica del Instituto Politécnico Nacional (IPN) de México en 1975. Es maestro en ciencias con especialidad en Control por el CINVESTAV- IPN en México 1986. Es también maestro en ciencias en sistemas de la manufactura con especialidad en Robótica por el ITESM–UT (USA) en 1993. También obtuvo el grado de Doctor en Ingeniería y Ciencias Aplicadas en Cuernavaca Morelos, México en 2012 por el CIICAP-UAEM. Sus intereses de investigación son Programación Matemática aplicada a la Robótica Humanoide, Control Automático y Redes Neuronales – Deep Learning.

José Alfredo Hernández-Pérez received a Professional Diploma in chemical engineering from Veracruzana University (Veracruz). He received an M.S. degree in food science from Instituto Tecnológico de Veracruz (Veracruz) and a Ph.D. degree in process engineering from École Nationale Supérieure des Industries Agricoles et Alimentaires (Paris, France). He works mainly on artificial intelligence in engineering processes. His research interests include modeling and simulation processes, optimization, and state estimation with application in heat and mass transfer processes and image analysis. He has published more than 110 articles. He is also a Reviewer for Neurocomputing, Energy, International Journal of Heat and Mass Transfer, International Journal of Thermal Sciences, International Journal of Refrigeration, Journal of Food Engineering, JAFC, MPE, Desalination, WASJ, RMCG, CABEQ, IJACT, Arabian JSE, Desalination and Water Treatment, IJEIS, IJTS, LAAR, and others. Finally, he is a member of the editor board of Computational Intelligence and Neuroscience (Impact Factor 2.28 according to the 2020 Journal Citation Reports released by Clarivate Analytics in 2018).

Prof. G. Nagarajan works at Internal Combustion Engine at Anna University, Chennai, 600-025 INDIA. His expertise covers different types of engines for car manufacture, from internal combustion up to electrical engines. His research also deals with the efficiency of energy cycles. He has developed projects with the auto-manufacturer Land rover. Expertise: Automobile Engineering. I.C Engines, Thermal related areas. Recognitions: Paper titled "Enhancing the Wear Resistance of Case Carburized Steel (En 353) by Cryogenic Treatment" published in Cryogenics Journal was listed in the TOP 25 Hottest Articles, January -March 2006. Paper titled "An Experimental Investigation on DI Diesel Engine with Hydrogen Fuel" published in Renewable Energy Journal was listed in the TOP 25 Hottest Articles, January - March 2008. Paper titled "Performance, Emission and Combustion Studies of a DI Diesel Engine Using Distilled Tire Pyrolysis Oil-Diesel Blends" published in Fuel Processing Technology Journal was listed in the TOP 25 Hottest Articles, January -March 2008.

David Juárez-Romero realizó su licenciatura en Ingeniería Química en la Fac. Química-UNAM, y sus estudios de maestría y doctorado en el Colegio Imperial de la Universidad de Londres, U. K. Su línea de investigación es “la mejora del Diseño y la operación de procesos de separación-transformación relacionados con máquinas de energía”. Desarrolla metodologías para analizar, diseñar, y controlar estos procesos. Pertenece al Sistema Nacional de Investigadores. Computación (2009), y Maestro en Ciencias de la Computación (2008) por la Universidad Autónoma de Aguascalientes (UAA). Graduado de la Licenciatura en Informática por el Instituto Tecnológico de Aguascalientes (2004).