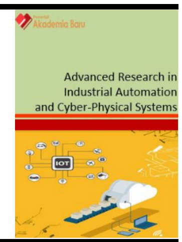
Penerbit Akademia Baru

Journal of Advanced Research in Industrial Automation and Cyber-Physical System

Journal homepage: www.akademiabaru.com/ard.html

ISSN: 2637-0263



Manipulating Gene Expression for Data Batch Assignment

Open Access

Ahmad Yusairi Bani Hashim[1,*]

[1] Faculty of Manufacturing Engineering, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia

ARTICLE INFO

Article history:
Received 25 July 2018
Received in revised form 7 August 2018
Accepted 17 August 2018
Available online 23 August 2018

Keywords:
Data storage, data retrieval, transcription, translation, receptor protein, gene expression

ABSTRACT

Gene expression is the flow of genetic information from deoxyribonucleic acid via ribonucleic acid to creating a receptor protein. We explain how we manipulate the mechanics of gene expression on some known applications such as in control systems and neural networks. The defined usage of the applications was simple and straightforward. The results of the manipulation of gene expression onto those applications were promising. The idea is to have a mechanism that extracts data from a system, calculate and find the outcomes, which are stored and may be retrieved when necessary. Depending on how the instructions are written, such data may be stored in the computer memory. The data may be retrieved from the memory with respective addresses.

1. Introduction

The classical methods in artificial intelligence such as genetic algorithm and neural network mimic the biological chromosomes and neural circuits in their pseudocode. These methods are modeled as such they function somewhat resemble the working processes found in the biological system. For example, imitating a neuron where a receptor protein permits data transmission in a network [1]. Due to the approach of copycatting systems in the natural world, the methods fall under a general field known as the evolutionary computation.

We saw researchers apply neural networks and genetic algorithm in solving real-world problems such as [2-4]. The solutions to the issues showed promising results. Using genetic programming, for instance, the creation of offspring that represent the developed passcode for a security goal was proposed in [5]. Of all the techniques in artificial intelligence, the mechanics of the protein production, and how it may be evolutionarily computed is absent.

As the algorithm is learning, a very large amount of data is being produced. It is referred to as big data that the processor is dumping into the local memory or in the cloud for storage. The data is

* *Corresponding author.*
E-mail address: yusairi@utem.edu.my (Ahmad Yusairi Bani Hashim)

stored in some locations determined by the default program. To retrieve the data, a complicated mining procedure has to be devised and executed.

As such, we seek to expand the procedure in evolutionary computation where the idea how the massively computed data is stored and retrieved is then made crisp. It is the issue of the moment when the processor needs to read and write the memory but lacking confined process because a default program has arranged the stacking of the data. It is the undertaking of this work to outline the sending and receiving of data to the memory according to the specified batch assignment. The aim is to assign specific data that is classified as the proteins.

This paper is organized as follows: the first part introduces gene expression followed by the second part that defines a gene expression model. Lastly, it deals with the manipulation of gene expression on transfer functions and a learning algorithm.

2. On Gene Expression

In a human body, there is a substantial number of cells. There is a specific job associated with each of these cells. In a cell, the diversity of proteins orchestrates as a tiny machine that operates it. Individual cells that are arranged end-to-end make up the nerve networks. Unique proteins on the surface of the receiving end of a cell are known as the receptor protein. These proteins are responsible for picking up the signals and passing it along to the next cell.

The nerve cells have grown branches, which help them communicate with their neighbors in the line. Particular proteins called structural proteins help cells extend these branches and hold them in place. These proteins are like bricks, stacking together to form column-like supports that give the cell its shape. For producing proteins, the cells use the information encoded in their genes.

Each gene in the DNA encodes information about how to make an individual protein. When a cell needs to make a specific protein, specialized machinery within the cell's nucleus interprets the gene and then exploits that information to produce a molecular message in the form of RNA (ribonucleic acid). RNA travels into the cytoplasm from the nucleus of the cell. This process is called the transcription of DNA produces mRNA (messenger) or merely the RNA.

RNA is sent to the protein-making machinery, the ribosome where the translation process takes place. Ribosome deciphers the message and makes a protein exactly matches the specifications laid out in the gene. Once established, the protein travels to the part of cells where it is needed and begins to work. The sequence of these processes explains the flow of genetic information from DNA via RNA to a receptor protein, hence gene expression (GE) [6]. Gene expression, on the other hand, establishes hard-wired pathways according to the environment where cells adapt to familiar changes in their environment [7].

3. Models

3.1 Definition

The components in GE: DNA ($\mathbb{d}$), mRNA ($\mathbb{m}$), ribosome ($\mathbb{r}$), and protein ($\mathbb{p}$) are expressed as a model for computional purposes.

3.2 Theorem

$$\{\mathbb{m} \ni \mathbb{d}\} \rightarrow \{\mathbb{r} \Vdash \mathbb{p}\} \qquad (1)$$

3.2.1 Proof

A DNA contains complete information of a living organism. Similarly, a transfer function includes data of the block, hence $\mathbb{d}$. Transcription of $\mathbb{d}$ extracts $\mathbb{m}$ for $\mathbb{r}$ leads to $\mathbb{p}$.

3.3 Modeling GE on a transfer function
3.3.1 Corollary

$$\{\mathbb{d} \equiv G(s), \mathbb{m} \equiv \{K, a, b\}\} \to \{\mathbb{r} \equiv \mathcal{L}^{-1}(G(s)|_{\mathbb{m}}), \mathbb{p} \equiv \mathbb{r}(t)\} \tag{2}$$

3.3.1 Proof

Let a simple transfer function be $G(s) = C(s)/R(s)$ with gain K, the roots of the denominator, a and b [8]. Using the element of $\mathbb{m}$ in the translation, that is the inverse laplace for $\mathbb{r}$ leads to $\mathbb{p}$. See Figure 1 for a clear description. Suppose a transfer function is given as

$$\mathbb{d} \equiv \frac{K}{s(s^2+as+b)} \tag{3}$$

Thus, transcription of (1) gives

$$\mathbb{m} \equiv \{K, a, b\} \tag{4}$$

Translation of (2) yields

$$\mathbb{r} \equiv \mathcal{L}^{-1}\left(\frac{K}{s(s^2+as+b)}\right)|_{\mathbb{m}} \tag{5}$$

$$\mathbb{p} \equiv K\left(\frac{1}{b} - \frac{1}{2bx}\left(a\left(-e^{t\left(-y-\frac{a}{2}\right)}\right)\right) + ae^{t\left(y-\frac{a}{2}\right)} + xe^{t\left(-y-\frac{a}{2}\right)} + xe^{t\left(-y-\frac{a}{2}\right)} + xe^{t\left(y-\frac{a}{2}\right)}\right) \tag{6}$$

$$x = \sqrt{a^2 - 4b}\,, y = \frac{x}{2} \tag{7}$$

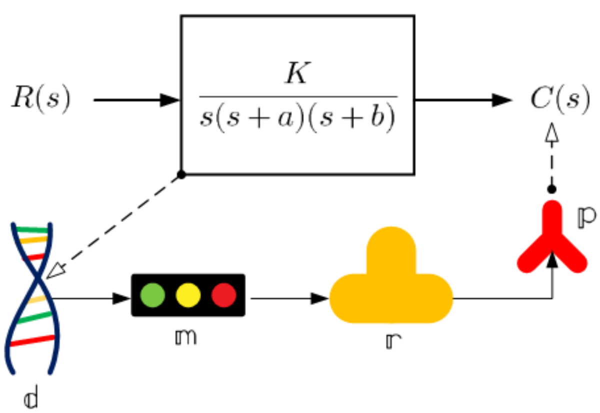


Fig. 1. Harmonizing GE with a transfer function

3.4 Modeling GE on Widrow-Hoff Learning Algorithm

The Widrow-Hoff learning algorithm is a classical neural network that functions in learning with supervision approach [9]. We present a single perceptron network where Widrow-Hoff learning rule is applied. Recent applications of the algorithm includes facial recognition [10] where the algorithm learns from the dynamics of pixel variables; tracking attributes where the algorithm learns from the intricate patterns in association with bioinformatics [11]; monitoring diesel fuel [12] where the algorithm learns the diesel fuel properties based on the near-infrared spectrum received from the transducers.

3.4.1 Definition

Suppose a network has a weight vector, $\mathbf{w}$; input vectors, $\mathbf{x}$; a learning constant, η; a desired output vector, $\mathbf{d}$; and an actual output vector, $\mathbf{o}$. On every iteration, n the weights will be updated. The iteration will stop when the sum of squared error approaches zero.

$$\Delta w^n = \eta(d^n - o^n)x^n, n = 1, \cdots, m \tag{8}$$
$$o^n = \sum w^n x^n \tag{9}$$
$$w^{n+1} = \Delta w^n + w^n \tag{10}$$

3.4.2 Corollary

$$\left\{\mathbb{d} \equiv \{\mathbf{x}, \eta, d, w\}, \mathbb{m} \equiv \{\eta, d\}\right\} \rightarrow \{\mathbb{r} \equiv \Delta w^n, \mathbb{p} \equiv w^{n+1}\} \tag{11}$$

3.4.3 Proof

Let $\mathbb{d}$ represents the network parameters: x, η, d, w. In transcription, only the learning constant and the desired output are extracted. In translation, the change of weight is being computed. At the conclusion of the process, the updated weight vector is produced, hence $\mathbb{p}$. See Figure 2 for a deeper explanation.

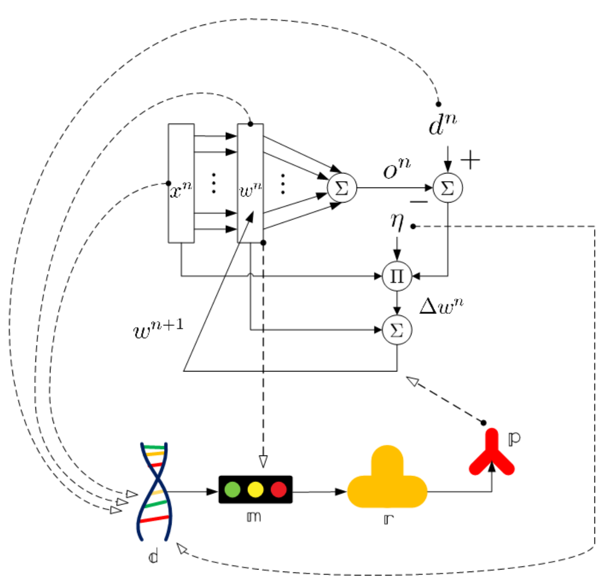


Fig. 2. Blending GE with Widrow-Hoff learning rule

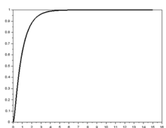
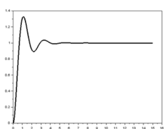
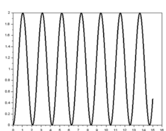
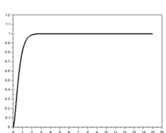
4. Manipulation

4.1 Manipulating GE on a Transfer Function

Table 1 lists, based on Eq. 2 to Eq. 6, several plots of step responses using the different combination in m. The plots were created using Scilab 5.5.2 [13]. In row 1, with $\mathbb{m}_1 \equiv \{9,9,9\}$ resulted in overdamped with poles $(-1.146, -7.854)$. Generally, using diverse combinations of elements in $\mathbb{m}_i$ shall produce a distinct $\mathbb{p}_i$, respectively. Now, the response shapes represent the $\mathbb{p}_i$ that somewhat depict if the system is stable, unstable, or marginally stable.

Depending on how the instructions are written, such data may be stored in random access memory or read-only memory. The data may be retrieved from the stacked memory with respective addresses. With the data presented above, $\mathbb{m}_1 \equiv \{\$9, \$9, \$9\}$ resulted in overdamped with poles ($-1.25604189374BC6A7EF9E, $-7.DA9FBE76C8B439581062), stored in hexadecimal (HEX).

Table 1
The protein plots

i	$\mathbb{m}_i$	$\mathbb{p}_i$	Response	Data in stacked memory in assembly code (HEX)
1	{9,9,9}		Overdamped	$9 $9 $9
2	{9,2,9}		Underdamped	$9 $9 $9
3	{9.0.9}		Damped	$9 $0 $9
4	{9,5.9,9}		Critically damped	$9 $5.E6666 $9

4.2 Manipulating GE on Antenna Control System

We now take an antenna control system (ACS) to examine if our GE model applicable in determining the stability of the system [14]. Equation 11 is the transfer function for the ACS. The task is to find the range of preamplifier gain K required to keep the closed-loop system stable.

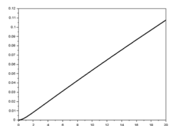
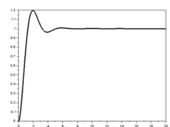
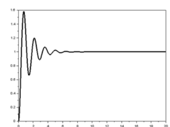
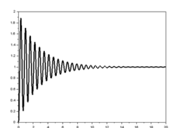
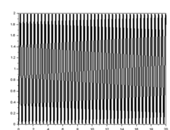
$$T(s) = \frac{6.63K}{s^3 + 101.7s^2 + 171s + 6.63K} \qquad (12)$$

Using Routh table approach as shown in Table 2, the system is found stable for $0 < K < 2623$. Alternatively, we use GE model to find the required K. We follow the process as described above and letting $6.63K = c$, where c is one of the roots of the denominator.

The results from Table 3 show that if we wish to have stable underdamped responses then, the value for preamplifier gain should be somewhere close to $K = 301.66$ and $\mathbb{m}_4 \equiv$

$\{301.66,107.71,171,2000\}$. This model also proves that the range of preamplifier gain for the system to stay stable is $0 < K < 2623$. In HEX, $\mathfrak{m}_4 \equiv$ {$12D.A8F5C28F5C28F5C28F5C, $6B.B5C28F5C28F5C28F5C29,$AB,$7DO.

Table 2
The protein plots for ACS

j	$\mathfrak{m}_j \equiv \{K, 107.71, 171, c\}$	$\mathfrak{p}_j$	Data in stacked memory in assembly code (HEX)
1	$\{0.1508, a, b, 0.998\}$		$0.269AD42C3C9EECBFB15B $6B.B5C28F5C28F5C28F5C29 $AB $0.FF7CED916872B020C49C
2	$\{50, a, b, 331.50\}$		$32 $6B.B5C28F5C28F5C28F5C29 $AB $14B.8
3	$\{301.66, a, b, 2000\}$		$12D.A8F5C28F5C28F5C28F5C $6B.B5C28F5C28F5C28F5C29 $AB $7DO
4	$\{1508.30, a, b, 10000\}$		$ 5E4.4CCCCCCCCCCCCCCCCCCD $6B.B5C28F5C28F5C28F5C29 $AB $2710
5	$\{2623, a, b, 17390.49\}$		$A3F $6B.B5C28F5C28F5C28F5C29 $AB $43EE.7D70A3D70A3D70A3D70A

4.3 Manipulation GE on Widrow-Hoff Learning Algorithm

Suppose a single neuron that has four inputs and four initial weights as seen in Figure 2 [15]. The initial input vectors, an initial weight vector, a learning rate, and the desired output vector are defined in Eq. 13. The parameters' superscript represents the iteration number.

$$x^1 = \begin{pmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{pmatrix}, x^2 = \begin{pmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{pmatrix}, x^3 = \begin{pmatrix} 0 \\ 1 \\ -1 \\ 1.5 \end{pmatrix}, w^1 = \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{pmatrix}, \eta = 1, d = (1\ -1\ 0) \tag{13}$$

Using the information in Eq. 13 and the definitions in Eq. 8 to Eq. 10, the respective parameters are found in Eq. 14a, Eq. 14b, and Eq. 14c. The process may be recurring to obtain the next iteration outcomes. In an actual neural network runs, it takes hundreds or more iterations to find the optimum weights. For simplicity, we show the results after the second iteration. In fact, Eq. 14a, Eq. 14b, and Eq. 14c are outcomes from the first iteration.

$$o^1 = w^{1,T} x^1 = 3 \tag{14a}$$

$$\Delta w^1 = \eta(d^1 - o^1)x^1 = \begin{pmatrix} -2 \\ 4 \\ -3 \\ 0 \end{pmatrix} \quad (14b)$$

$$w^2 = w^1 + \Delta w^1 = \begin{pmatrix} -1 \\ 3 \\ -3 \\ 0.5 \end{pmatrix} \quad (14c)$$

Table 3 lists $\mathbb{m}_k$, $\mathbb{p}_k$, and the error. The proteins represent the updated weight after every iteration. The computed error of -3.75 leads to $\mathbb{p}_2$. Using $\mathbb{p}_2$, on the other hand, leads to an error of -9.56. Thus, there is a distinct relationship between the protein and the error, which is recorded on every iteration.

Table 3
The protein synthesis for the network

n	k	$\mathbb{m}_k$	$\mathbb{p}_k$	error $= d - o$	Data in stacked memory in assembly code (HEX)
0	1	$\left\{1,1,\begin{bmatrix} 1 \\ -2 \\ 1.5 \\ 0 \end{bmatrix}\right\}$	$\begin{bmatrix} 1 \\ -1 \\ 0 \\ 0.5 \end{bmatrix}$	-2.00	\$1 \$1 \$1,\$-2,\$1.8,\$0 \$1,\$-1,\$0,\$0.8 \$-2
1	2	$\left\{1,-1,\begin{bmatrix} 1 \\ -0.5 \\ -2 \\ -1.5 \end{bmatrix}\right\}$	$\begin{bmatrix} -1 \\ 3 \\ -3 \\ 0.5 \end{bmatrix}$	-3.75	\$1 \$-1 \$1,\$-0.8,\$-2,\$-1.8 \$-1,\$3,\$-3,\$0.8 \$-3.C
2	3	$\left\{1,0,\begin{bmatrix} 0 \\ 1 \\ -1 \\ 0.5 \end{bmatrix}\right\}$	$\begin{bmatrix} -4.75 \\ 4.88 \\ 4.50 \\ 6.13 \end{bmatrix}$	-9.56	\$1 \$0 \$0,\$1,-1,\$0.8 \$-4.C,\$ 4.E147AE147AE147AE147B,\$4.8,\$ 6.2147AE147AE147AE147 \$-9.8F5C28F5C28F5C28F5C3

Based on the data above, the recorded data is stored as \$-3.C and \$-9.8F5C28F5C28F5C28F5C3. When reading the data, the processor reads the memory by first sending out a coded signal on the address bus. After a certain delay, the memory places the contents of the selected memory location on the address bus. A similar fashion occurs when writing data.

5. Conclusions

The mechanics of the receptor proteins creation is significant that this study that proposes an algorithm for computer data storage and retrieval, as a prove-of-concept. The philosophy that lies in this model is the combination of three processes: transcription, translation, and protein creation. It turns out that by compiling all information about a system, extracting some of the vital information may lead to the clear solution when computed. In doing so, the extracted information and the respective solution can be stored as history that can be retrieved when needed. Depending on how the instructions are written, such data may be stored in the computer memory. The data may be retrieved from the memory with respective addresses.

References

[1] Destexhe, A., Z.F. Mainen, and T.J. Sejnowski, Kinetic models of synaptic transmission. Methods in neuronal modeling, 1998. 2: p. 1-25.
[2] Gruau, F., Neural Network Synthesis using Cellular Encoding and the Genetic Algorithm. 1994.
[3] Kumar, M. and D.P. Garg. Intelligent learning of fuzzy logic controllers via neural network and genetic algorithm. in Proceedings of. 2004.
[4] Chaiyaratana, N. and A.M. Zalzala, Time-optimal path planning and control using neural networks and a genetic algorithm. International Journal of computational intelligence and applications, 2002. 2(02): p. 153-172.
[5] Hashim, A.Y.B., A Scheme to Passcode Generation. International Journal of Information Security Science, 2018. 7(1): p. 1-10.
[6] Cullen, P., et al., Basic Concepts of Gene Expression, in Analysing Gene Expression, S. Lorkowski and P. Cullen, Editors. 2004, Wiley Online Library.
[7] Freddolino, P.L., et al., Stochastic tuning of gene expression enables cellular adaptation in the absence of pre-existing regulatory circuitry. eLife, 2018. 7: p. e31867.
[8] Palm, W.J., Control systems engineering. 1988: John Wiley & Sons, Inc.
[9] Widrow, B. and M.E. Hoff. Adaptive switching circuits. in Neurocomputing: foundations of research. 1988. MIT Press.
[10] Jannah, M., M. Zarlis, and H. Mawengkang. Facial expression system on video using Widrow-Hoff. in Journal of Physics: Conference Series. 2018. IOP Publishing.
[11] Urbanowicz, R.J., et al. Attribute tracking: strategies towards improved detection and characterization of complex associations. in Proceedings of the Genetic and Evolutionary Computation Conference. 2018. ACM.
[12] Al-kaf, H.A.G., K.S. Chia, and N.A.M. Alduais, A comparison between single layer and multilayer artificial neural networks in predicting diesel fuel properties using near infrared spectrum. Petroleum Science and Technology, 2018. 36(6): p. 411-418.
[13] Consortium, S., Scilab: The free software for numerical computation. Scilab Consortium, Paris, France, 2011.
[14] Nise, N.S., Control Systems Engineering. 2007: John Wiley & Sons.
[15] Siddique, N. and H. Adeli, Computational intelligence: synergies of fuzzy logic, neural networks and evolutionary computing. 2013: John Wiley & Sons.