FKAAB PROFESSIONAL TRAINING FOR EDUCATOR PROGRAMME

BASIC CONCEPT OF AI MACHINE/DEEP LEARNING

Speaker:

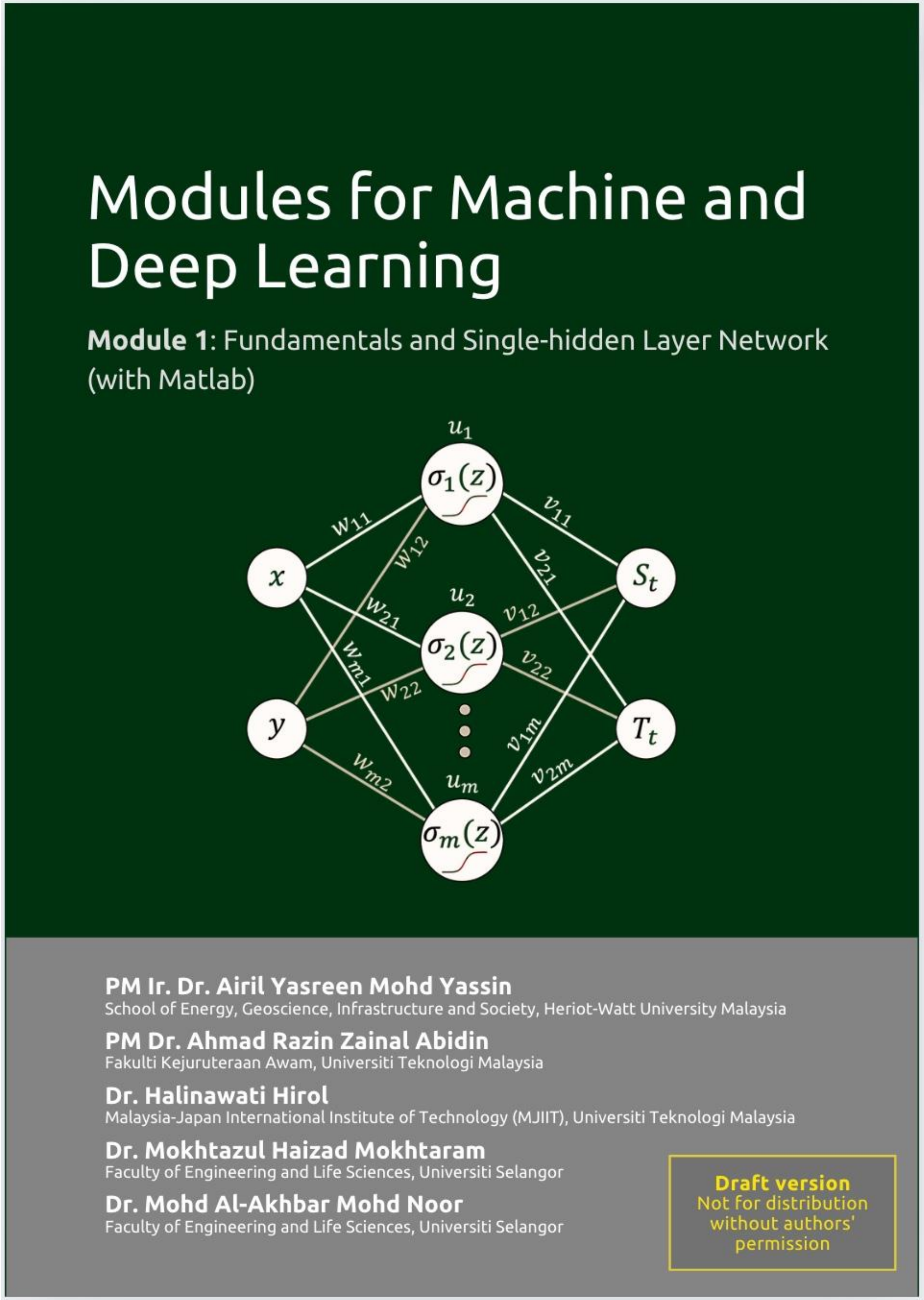
Associate Professor Ir Dr Airil Yasreen Mohd Yassin
School of Energy, Geoscience, Infrastructure and Society (EGIS)
Heriot-Watt University Malaysia

28 AUGUST 2024
10am – 1pm

Objectives of the talk:

1. To introduce the concept of AI machine/deep learning (how and why it works) from engineers' point of view

2. To familiarize the audiences (e.g. engineering students and educators) with terms and ideas of deep learning so that they can do self-study on the topic/field (through readings and by watching videos available on YouTube)



UTHM Universiti Tun Hussein Onn Malaysia

HERIOT WATT UNIVERSITY

UK | DUBAI | MALAYSIA

Please contact the secretariat to get a copy of this module.

It’s free!

A very quick introduction

Machine Learning (ML) is a subset of Artificial Intelligence (AI). In turn, Deep Learning (DL) is a subset of ML. And today's very hot topics of Generative AI and Large Language Model (LLM) are subsets of DL. This is as shown in Figure 1 [1].

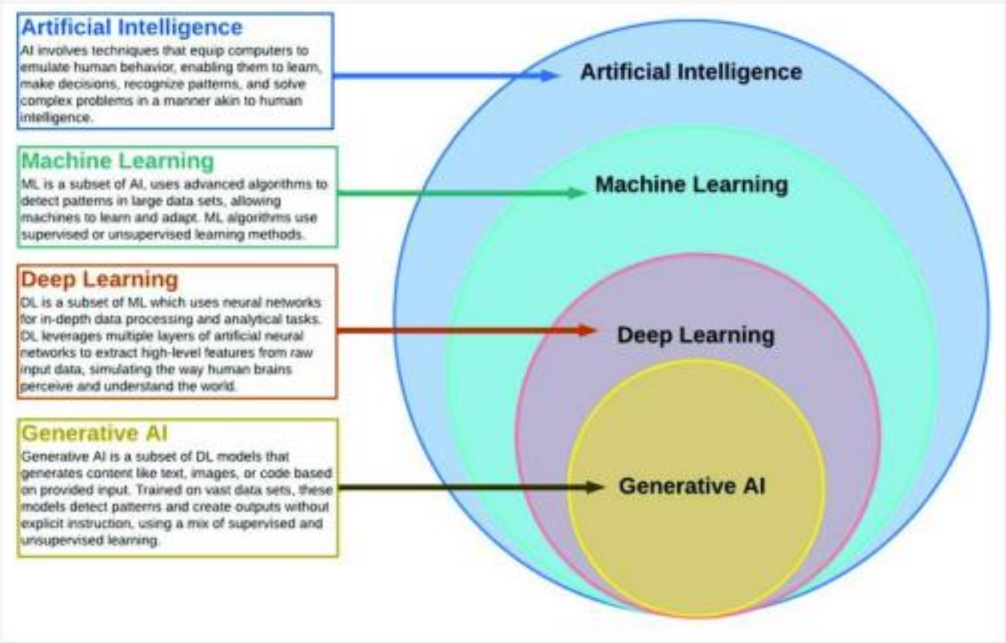


Figure 1: Specialization of deep learning (DL) in the field of AI [1]

DL is a type of ML that employs Artificial Neural-Network (ANN, or just NN) as its architecture.

Current technologies that use DL:

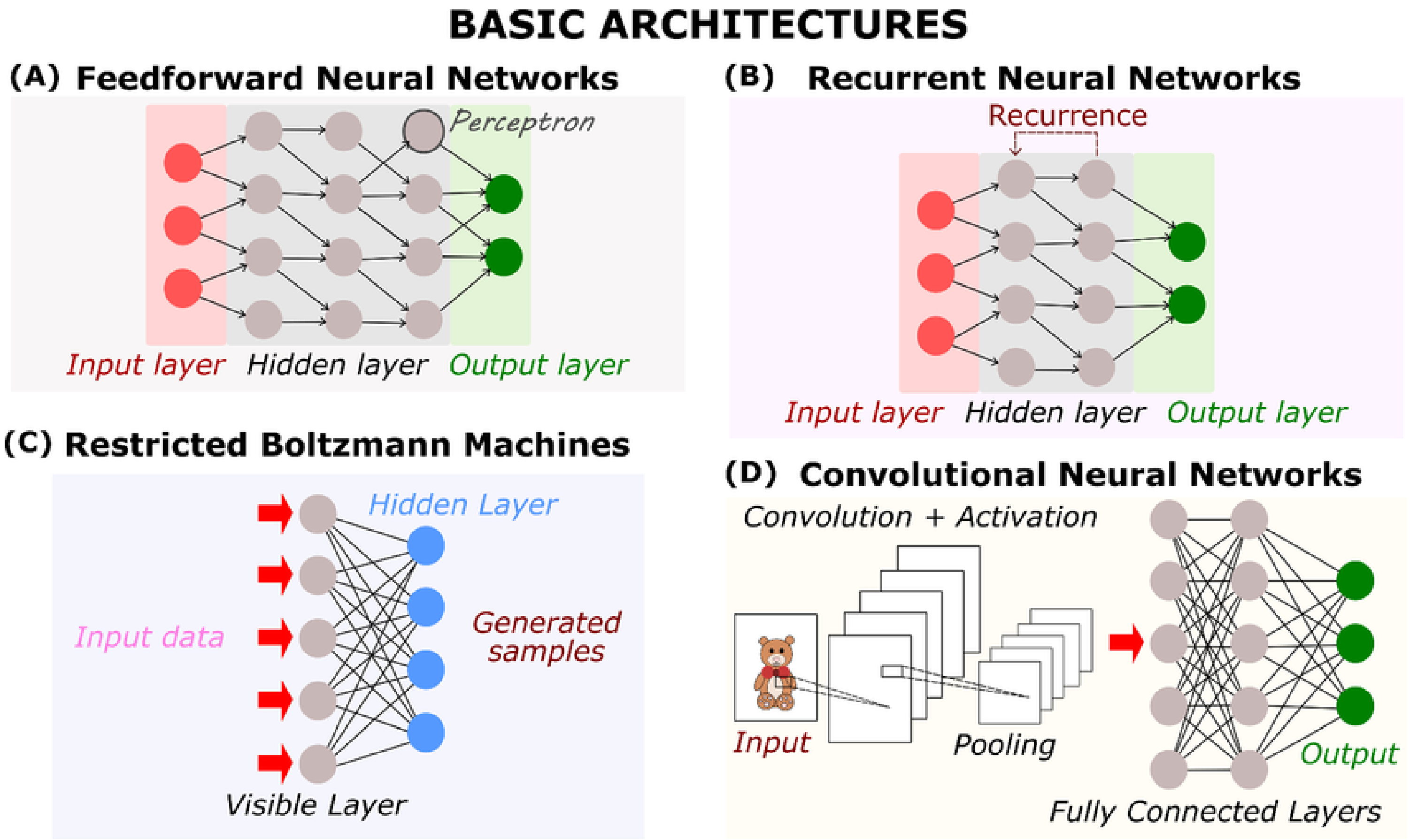
i) ChatGPT
ii) Google Translate
iii) Alpha Go
iv) many more

There are variants of DL, the three main ones are:

i) Feedforward Neural Networks (FFNNs)

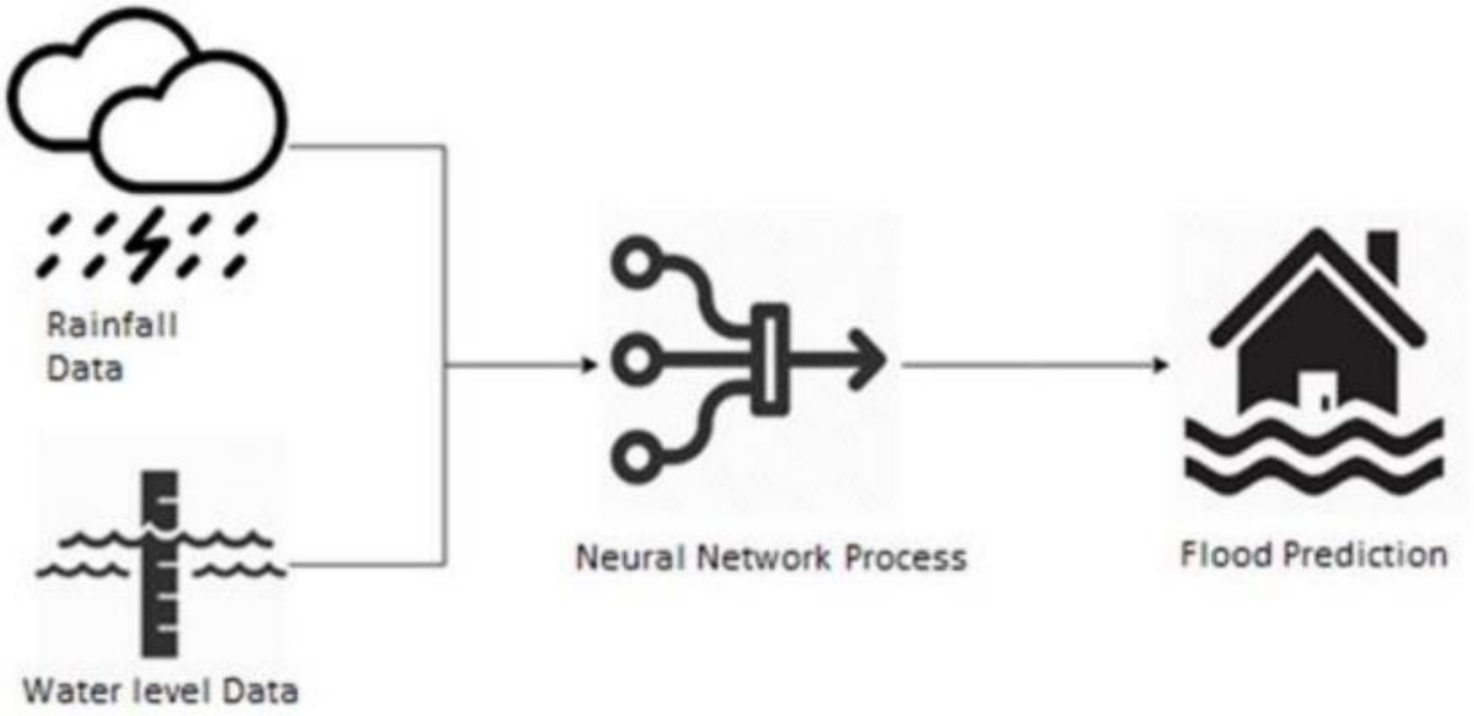
ii) Recurrence Neural Networks (RNNs)

iii) Convolutional Neural Networks (CNNs)



Ref: [4]

Examples of application of deep learning in Civil Engineering



ANN flood prediction system [2]

Table 2
Application of DL models for various hydrological and water resources applications.

Applications	References	DL Model used
Flood Forecasting	Hourly flood forecasting (Wu et al., 2018)	Context-aware LSTM with an attention mechanism
	Runoff prediction from short term extreme rainfall data (Li et al., 2021)	LSTM
	Flood forecasting (Ding et al., 2020)	LSTM with spatiotemporal attention mechanism
Weather Forecasting	(Zhang et al., 2022)	Spatiotemporal-LSTM with self-attention
	(Zhang et al., 2022)	CNN
	(Chen et al., 2019)	3-D CNN + LSTM
	(Giffard-Roisin et al., 2020)	CNN
	(Ham et al., 2019)	CNN
Streamflow prediction	(Saha et al., 2022)	CNN
	(Feng et al., 2020)	DI + LSTM
Soil moisture prediction	Short-term soil moisture forecasting (Li et al., 2022)	LSTM with an attention mechanism
	Soil moisture modeling (Fang et al., 2017)	LSTM
	Multilayer soil moisture estimation (Karthikeyan and Mishra, 2021)	XGBoost trained region-wise and layer wise
Rainfall-runoff modeling	Runoff estimation (Xiang et al., 2020)	Encoder-decoder LSTM
	Runoff estimation (Jiang et al., 2020)	Hybrid Physics-RNN and 1D-CNN
Water quality	Dissolved Oxygen level prediction (Zhi et al., 2021)	LSTM
	Short-term water quality prediction (Wan et al., 2022)	SOD-VGG-LSTM hybrid model
	Predicting spatiotemporal variations of Dissolved Oxygen levels (Yu et al., 2020)	DL model
	Water quality variables prediction (Bi et al., 2021)	LSTM-based encoder-decoder
	Water quality prediction (Bi et al., 2023)	Hybrid Encoder-decoder based BiLSTM with an attention mechanism
	River water quality (DO) prediction (Zhi et al., 2021)	LSTM
Water level prediction	Surrogate Water level prediction in Yangtze River (Pan et al., 2020)	CNN-GRU model
	Daily water level variation prediction (Xu et al., 2023a)	Transformer model

Ref: [5]

Examples of application of deep learning in Civil Engineering (authors' own work)

Applied Mathematics and Computational Intelligence
Volume 10, No.1, Dec 2021 [1-17]

UNIVERSITI MALAYSIA PERLIS

Damage Detection Formulation using Inverse Frequency Analysis incorporating Artificial Neural Network for Kirchhoff Plate Theory

M.H. Mokhtaram[1*], M.A. Mohd Noor[2], M.Z. Jamil Abd Nazir[2], D. Ahmad[1], M.K Marwah[1], H. Hamid[1], and A.Y. Mohd Yassin[3]
[1]Faculty of Engineering and Life Sciences, Universiti Selangor, Jalan Timur Tambahan, 45600 Bestari Jaya, Selangor, Malaysia.
[2]School of Civil Engineering, Faculty of Engineering, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia.
[3] School of Energy, Geoscience, Infrastructure and Society, Heriot-Watt University Malaysia, 62200 Putrajaya, Malaysia.

UTHM Universiti Tun Hussein Onn Malaysia HERIOT WATT UNIVERSITY
UK | DUBAI | MALAYSIA

Examples of application of deep learning in Civil Engineering (authors' own work)

NAHRIM HERIOT WATT UNIVERSITY UTM UNIVERSITI TEKNOLOGI MALAYSIA unisel UNIVERSITI SELANGOR

Perbincangan Awalan Pembangunan Prototaip Sistem Ramalan Banjir Berasaskan Kecerdasan Buatan (Artificial Intelligence/Machine Learning) antara NAHRIM, HWUM, UTM dan UNISEL

How are we going to conduct the workshop?

Answer:

We are going to jump straight to the action and discuss the details along the way.

Therefore, in this workshop, a simultaneous equation is immediately given, and our task is to formulate a neural network (and deep neural network, if time permits) that able to produce “similar” results. As we progress with our derivation we will introduce and discuss specific features when it prompts explanation.

The “unknown” equations to be mimic

In this workshop, we will create a network that is “equivalent” to the following equations:

$$3x + 2y = s \qquad (1a)$$

$$2x + 6y = t \qquad (1b)$$

or in matrix forms,

$$\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} s \\ t \end{Bmatrix} \qquad (1c)$$

But why we want to have these equations?

<u>Answer:</u>

to create the sets of “labelled data”, that is, the set of paired input (e.g. *s*, *t*) for the learning (e.g., training and testing) process of our network.

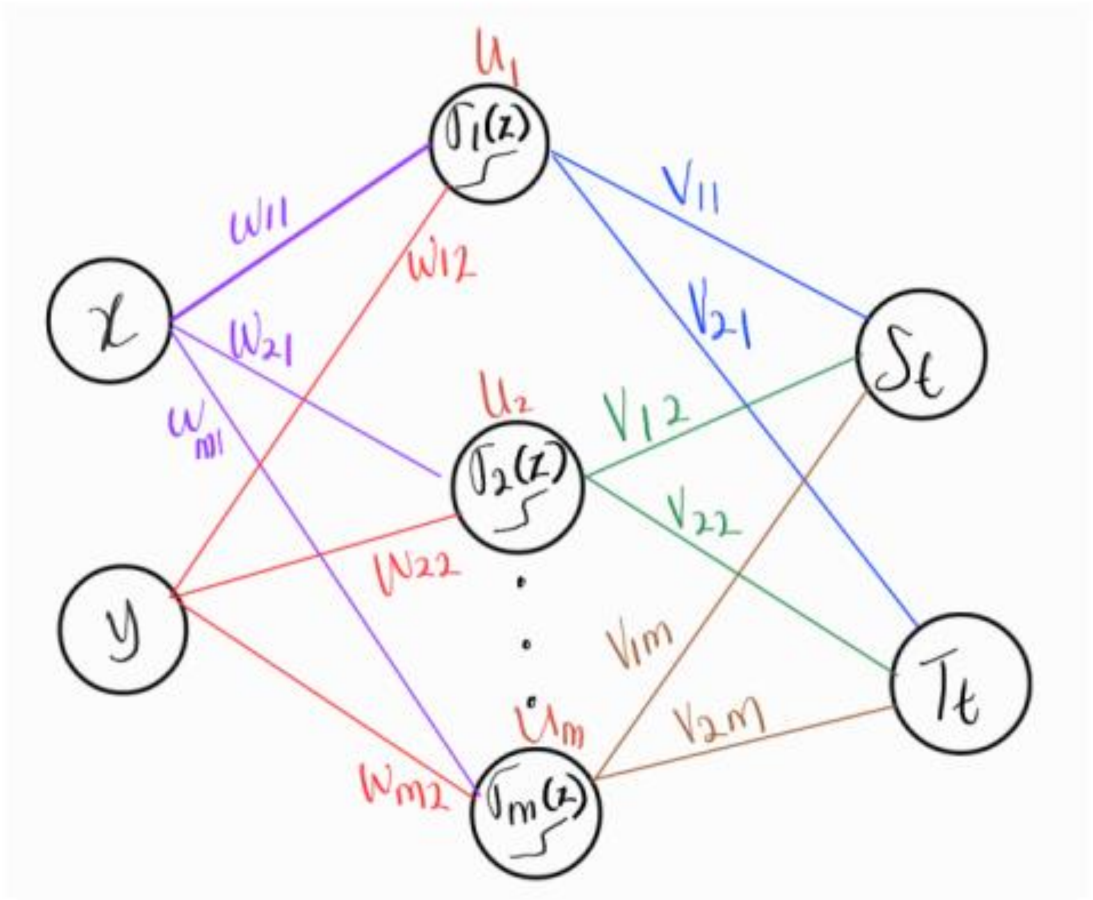
<u>Input</u>   <u>Actual Output</u>

$$\begin{Bmatrix} x_1, y_1 \\ x_2, y_2 \\ \vdots \\ x_n, y_n \end{Bmatrix} \xrightarrow{\text{Eq (1)}} \begin{Bmatrix} s_1, t_1 \\ s_2, t_2 \\ \vdots \\ s_n, t_n \end{Bmatrix}$$

Single hidden layer (shallow) neural network

To "mimic" Eqns. (1), we set a network below

Input layer Hidden layer Output layer



A single hidden layer (to "mimic" Eqns. (1))

Neurons

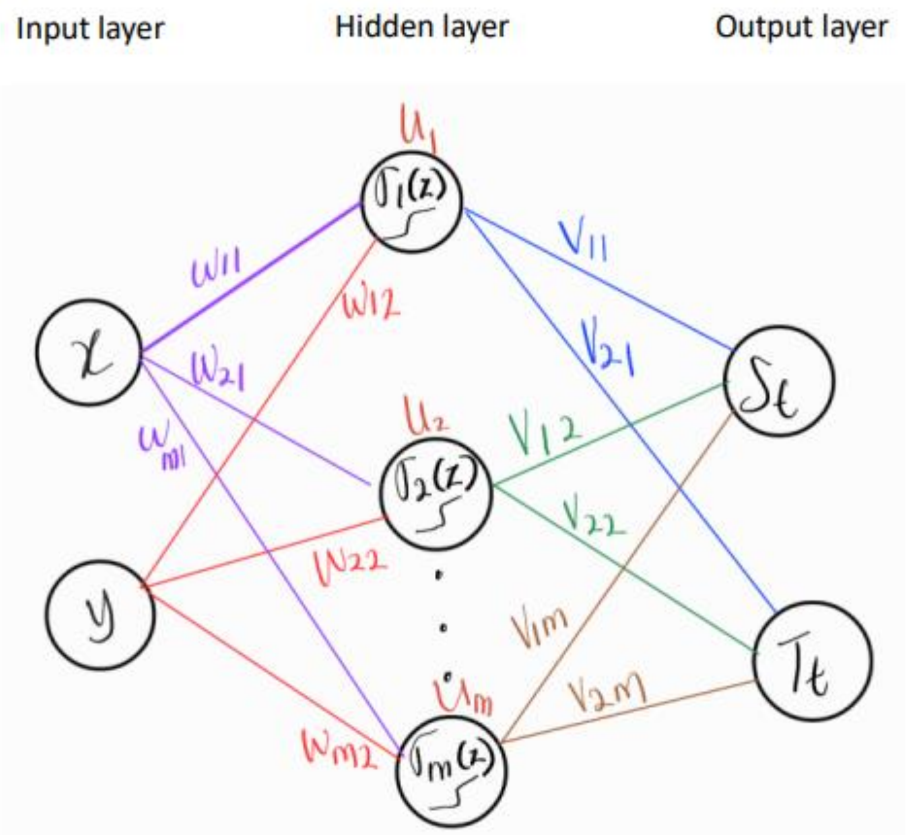
Except for circle circling input x and y, the circles symbolize (or called) neurons.

Input

$$\{\bar{x}\} = \begin{Bmatrix} x_k \\ y_k \end{Bmatrix} = \begin{Bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \end{Bmatrix} \qquad (2^*)$$

Channel

The lines connecting the circles (neurons) are called the channel. They guide the flow or the travel of "information" from one neuron to another.



Weights, w_{ji} and v_{ji}

$$[w] = \begin{array}{c} \text{Coming from} \\ \begin{bmatrix} w_{11} & w_{21} & \dots & w_{m1} \\ w_{12} & w_{22} & \dots & w_{m2} \end{bmatrix} \begin{array}{l} \leftarrow x \\ \leftarrow y \end{array} \end{array} \qquad (2)$$

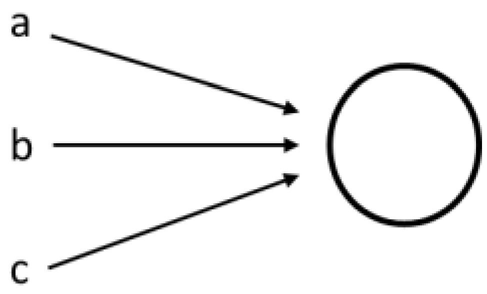
$$[v] = \begin{array}{c} \text{Going to} \\ \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1m} \\ v_{21} & v_{22} & \dots & v_{2m} \end{bmatrix} \begin{array}{l} \rightarrow \hat{s} \\ \rightarrow \hat{t} \end{array} \end{array} \qquad (3)$$

Biases, u_j

$$[u] = [u_1 \quad u_2 \quad \cdots \quad u_m] \tag{4}$$

Summing variable, z

$$z = a + b + c \tag{5}$$



Input into neuron

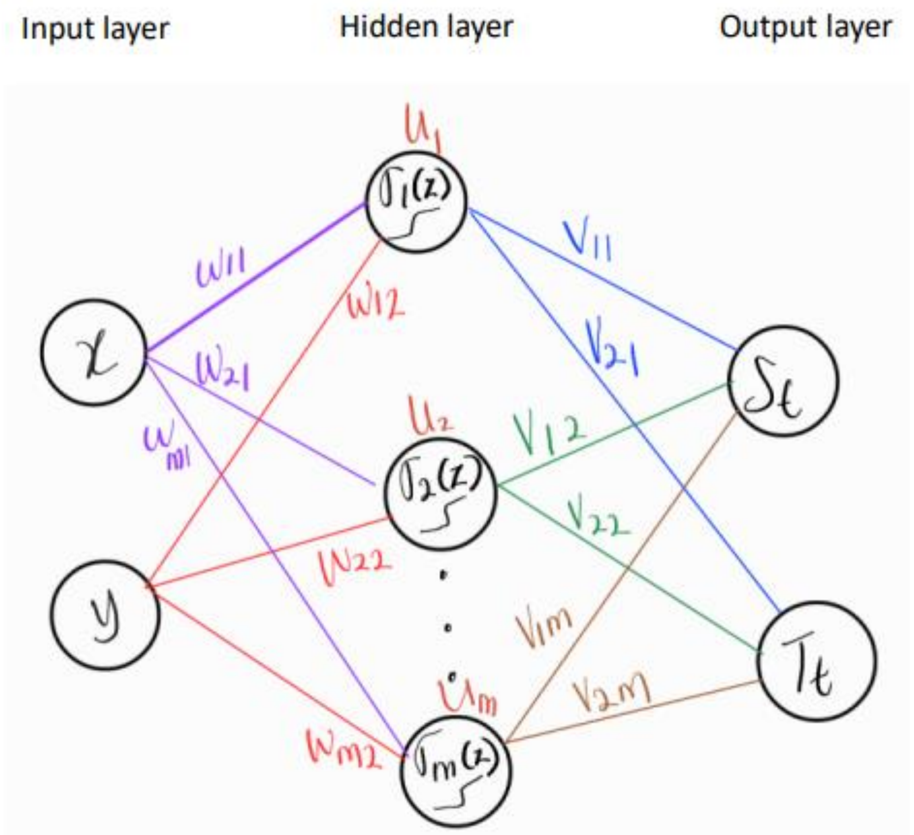
The activation functions, σ_j

$$\sigma_j = \frac{1}{1 + e^{-z_j}} \tag{6}$$

$$[\sigma] = [\sigma_1 \quad \sigma_2 \quad \cdots \quad \sigma_m] \tag{7a}$$

or

$$[\sigma] = \left[\frac{1}{1 + e^{-z_1}} \quad \frac{1}{1 + e^{-z_2}} \quad \cdots \quad \frac{1}{1 + e^{-z_m}}\right] \tag{7b}$$



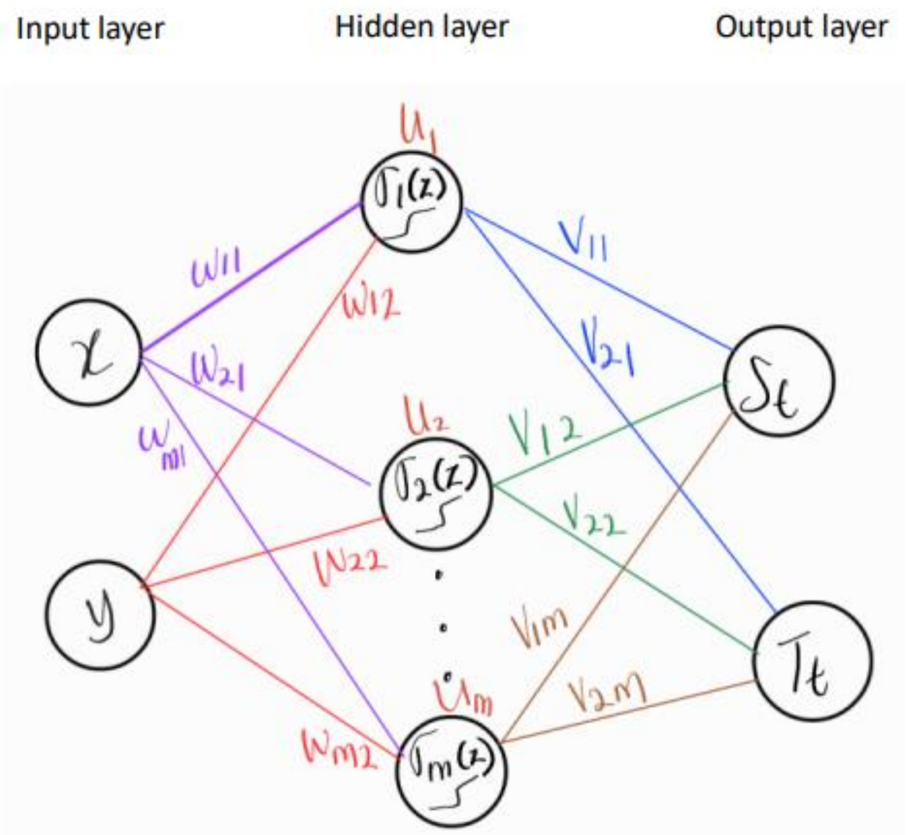
<u>The network output, S_t and T_t</u>

In Figure 4, S_t and T_t are the output and their neurons forms the output layer (the rightmost). What we want for these outputs to be as close as it can be to the correct ones. Specifically in our case to "mimic" Eq (1), we want S_t to be as close to s, and T_t to be as close to t. Once we able to do this, we can use the network to predict the output for the new or future input.

To get S_t and T_t to be as close as possible to s and t, respectively, we need to tune or train our weights and biases accordingly, a process we will detail later. Mathematically and by referring to Figure 4, we can state S_t and T_t as,

$$S_t = f(x, y, w, u, v) \tag{8a}$$

$$T_t = g(x, y, w, u, v) \tag{8b}$$



Training the network (allowing it to "learn")

Basically, there are four stages in building or completing an ML or a DL project:

i. Training
ii. Validation
iii. Testing
iv. Prediction/discovery (the ultimate use)

Training is the stage where we initialize the value of the parameters (e.g., weights and biases) then train or tune them iteratively (updating).

Validation would take place after the training. Validation involves the tuning of the hyperparameters such as learning rate, change of activation function, increase in the number of neurons and hidden layers etc. Validation might use the same sets of labelled data as in the training stage or different ones. The purpose of the validation is to check whether the performance of the network can be further optimized or be made more effective or/and economical.

Training the network (allowing it to "learn")

Basically, there are four stages in building or completing an ML or a DL project:

i. Training
ii. Validation
iii. Testing
iv. Prediction/discovery (the ultimate use)

Since stage ii, iii, and iv, are more a matter of implementation (rather than fundamentals or basic concepts), in this workshop, we focus on stage i, that is, on the training of the network.

In turn, there are two stages for the training of the network:

i. Forward-pass (calculating the network output)
ii. Backward-pass (backpropagation and updating)

Forward Pass (calculating the network output)

<u>Step 1: Multiplication</u>

Input x_k will be multiplied by weight w_{11} before goes into the first neuron through the channel. It will also be multiplied by weight w_{21} before goes into the second neuron, and so on. For input y_k, it will be multiplied by weight w_{12} before goes into the first neuron and so on.

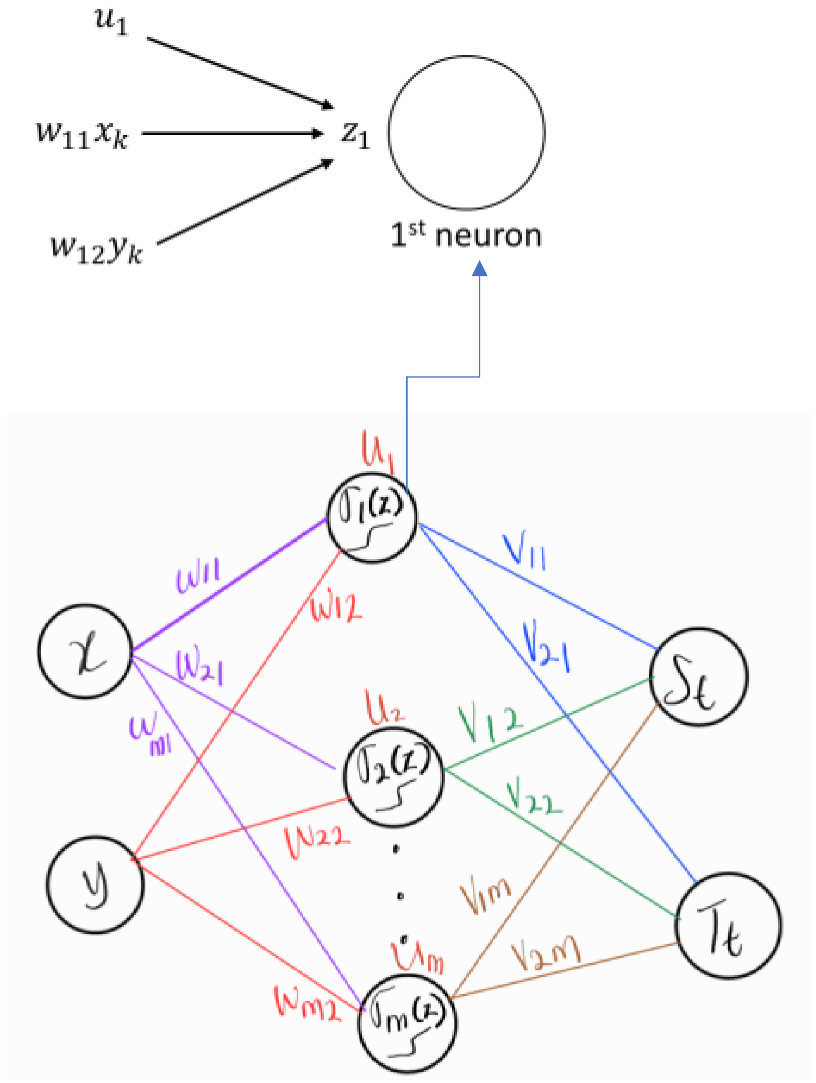
<u>Step 2: Summation</u>

But, before they enter the corresponding neuron, they will be summed first, then added by the corresponding bias. For example, for the 1st neuron, this will be

$$z_1 = w_{11}x_k + w_{12}y_k + u_1 \tag{9}$$

Whilst for the j^{th} neuron,

$$z_j = w_{j1}x_k + w_{j2}y_k + u_j \tag{10}$$



Forward Pass (calculating the network output)

Step 3: Nonlinearization

In the neuron, the summed input (e.g. z) will be nonlinearized by the activation function. For the first neuron, this is given as (from Eqns. (7) and (9)),

$$\sigma_1 = \frac{1}{1+e^{-z_1}}$$

or

$$\sigma_1 = \frac{1}{1+e^{-(w_{11}x_k + w_{12}y_k + u_1)}} \tag{11}$$

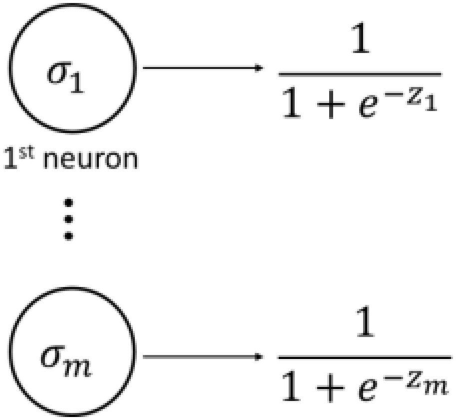
For the j-th neuron (from Eqns. (7) and (10)),

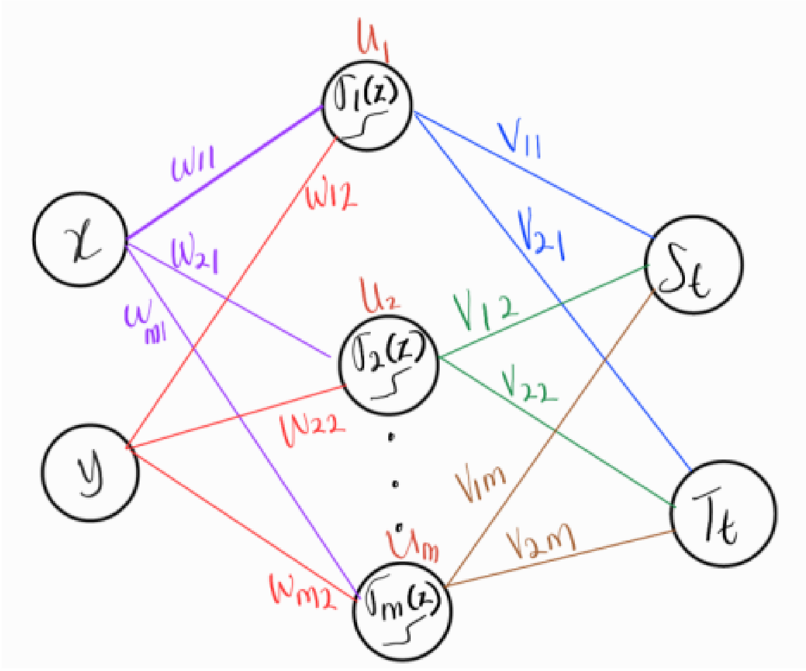
$$\sigma_j = \frac{1}{1+e^{-z_j}}$$

or

$$\sigma_j = \frac{1}{1+e^{-(w_{j1}x_k + w_{j2}y_k + u_j)}} \tag{12}$$

Note that, Eqns. (11) and (12) are also the output of the neurons in the hidden layer. This is graphically described by Figure 8.





Forward Pass (calculating the network output)

Step 4: Multiplication… again

For example, for the output of the first neuron, it will be multiplied by weight, v_{11} before it gets to output neuron, S_t. It will also be multiplied by v_{21} before it gets to output neuron, T_t. And so on.

Step 5: Summation and the calculation of network output, S_t and T_t

Finally, all the info (variables) will reach the output layer. They will reach S_t and T_t, accordingly. Usually, there will be no more nonlinearization in the output neuron, only summation, thus, (refer to Fig. 4);

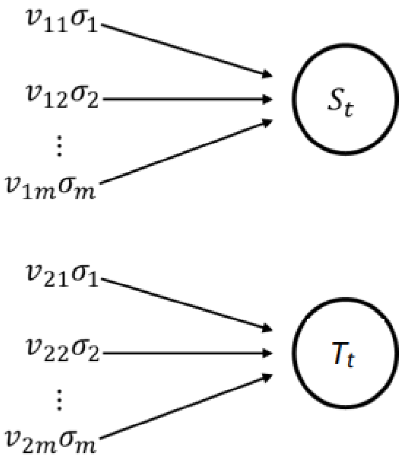
$$S_t = v_{11}\sigma_1 + v_{12}\sigma_2 \ldots + v_{1m}\sigma_m \tag{13}$$

$$T_t = v_{21}\sigma_1 + v_{22}\sigma_2 \ldots + v_{2m}\sigma_m \tag{14}$$

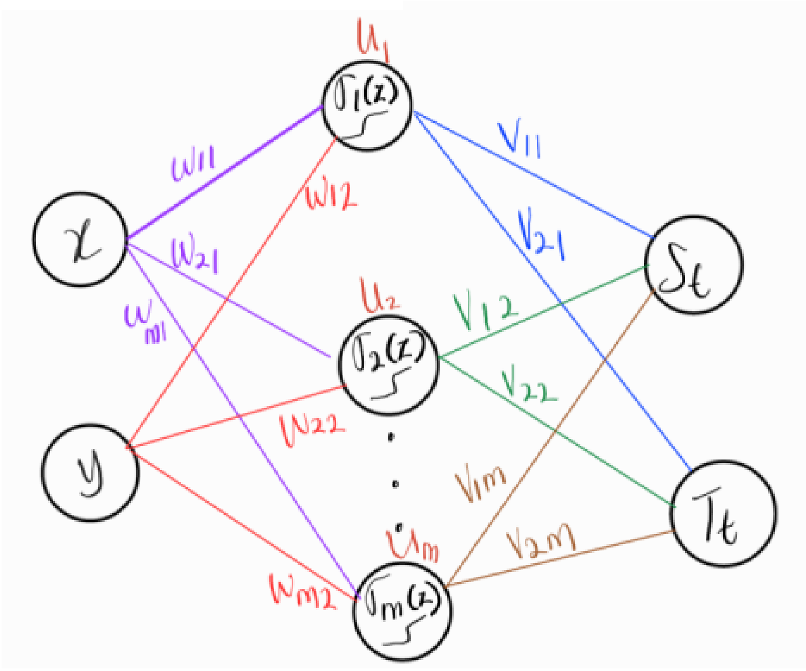
or (from Eqns. (11) and (12))

$$S_t = v_{11}\left(\frac{1}{1 + e^{-(w_{11}x_k + w_{12}y_k + U_1)}}\right) + \ldots + v_{1m}\left(\frac{1}{1 + e^{-(w_{m1}x_k + w_{m2}y_k + U_m)}}\right) \tag{15}$$

$$T_t = v_{21}\left(\frac{1}{1 + e^{-(w_{11}x_k + w_{12}y_k + U_1)}}\right) + \ldots + v_{2m}\left(\frac{1}{1 + e^{-(w_{m1}x_k + w_{m2}y_k + U_m)}}\right) \tag{16}$$



We have calculated the output!!!



Backward pass (backpropagation)

Errors

The errors specific for our network against Eqns. (1) can be given as

$$E_1 = (s - S_t) \tag{17a}$$

$$E_2 = (t - T_t) \tag{17b}$$

or from Eqns. (13) and (14) as,

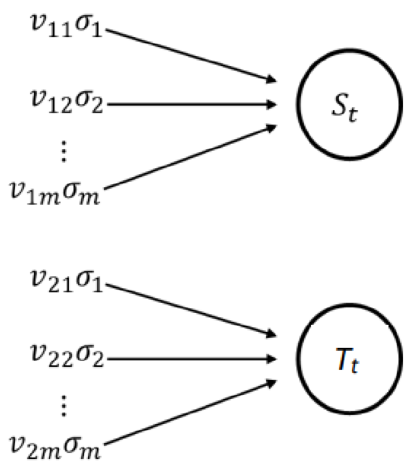
$$E_1 = (s - (v_{11}\sigma_1 + v_{12}\sigma_2 + \cdots + v_{1m}\sigma_m)) \tag{18a}$$

$$E_2 = (t - (v_{21}\sigma_1 + v_{22}\sigma_2 + \cdots + v_{2m}\sigma_m)) \tag{18b}$$

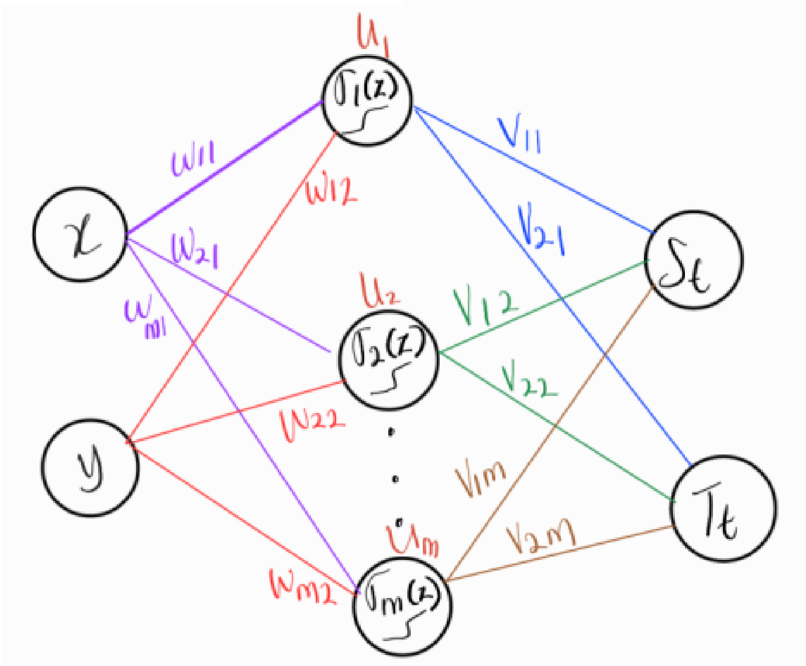
or from Eqns. (15) and (16) as,

$$E_1 = (s - (v_{11}(\frac{1}{1 + e^{-(w_{11}x_k + w_{12}y_k + u_1)}}) + \cdots + v_{1m}(\frac{1}{1 + e^{-(w_{m1}x_k + w_{m2}y_k + u_m)}}))) \tag{19a}$$

$$E_2 = (t - (v_{21}(\frac{1}{1 + e^{-(w_{11}x_k + w_{12}y_k + u_1)}}) + \cdots + v_{2m}(\frac{1}{1 + e^{-(w_{m1}x_k + w_{m2}y_k + u_m)}}))) \tag{19b}$$



We have calculated the output!!!



Backward pass (backpropagation)

Loss function and its minimization by gradient descent

Squared-Error Loss Function,

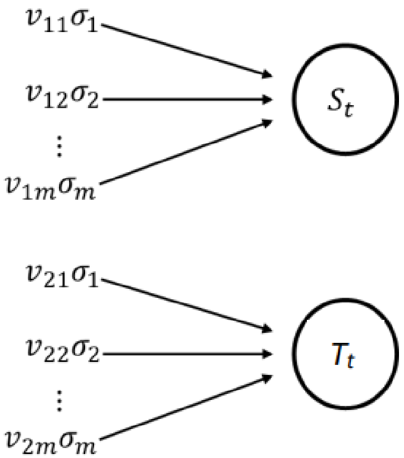
Loss function, $$L = {E_1}^2 + {E_2}^2 \tag{20a}$$

$$L = (s - S_t)^2 + (t - T_t)^2 \tag{20b}$$

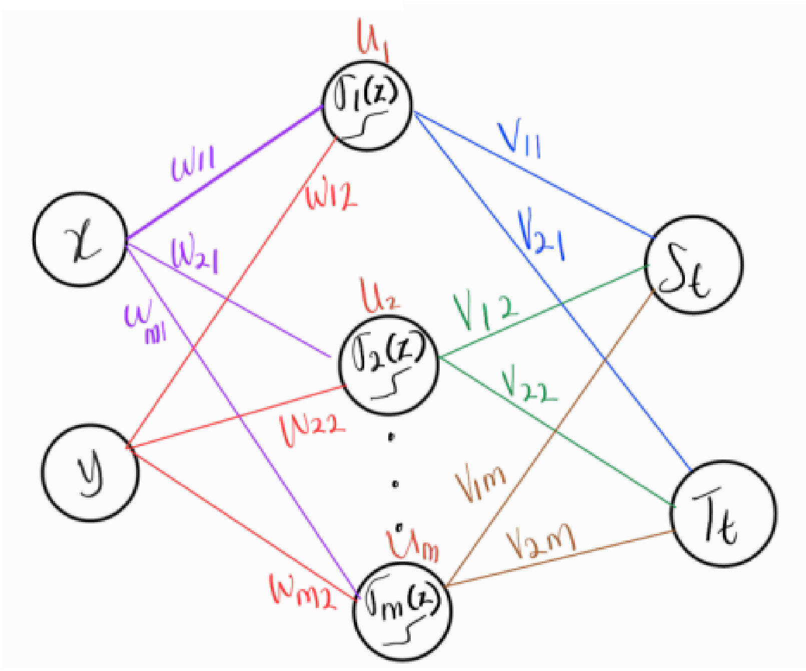
Mean-Squared

Loss function, $$L = \frac{1}{n}\sum_{k=1}^{n}({E_{1,k}}^2 + {E_{2,k}}^2) \tag{21a}$$

$$L = \frac{1}{n}\sum_{k=1}^{n}((s_k - S_{t,k})^2 + (t_k - T_{t,k})^2) \tag{21b}$$

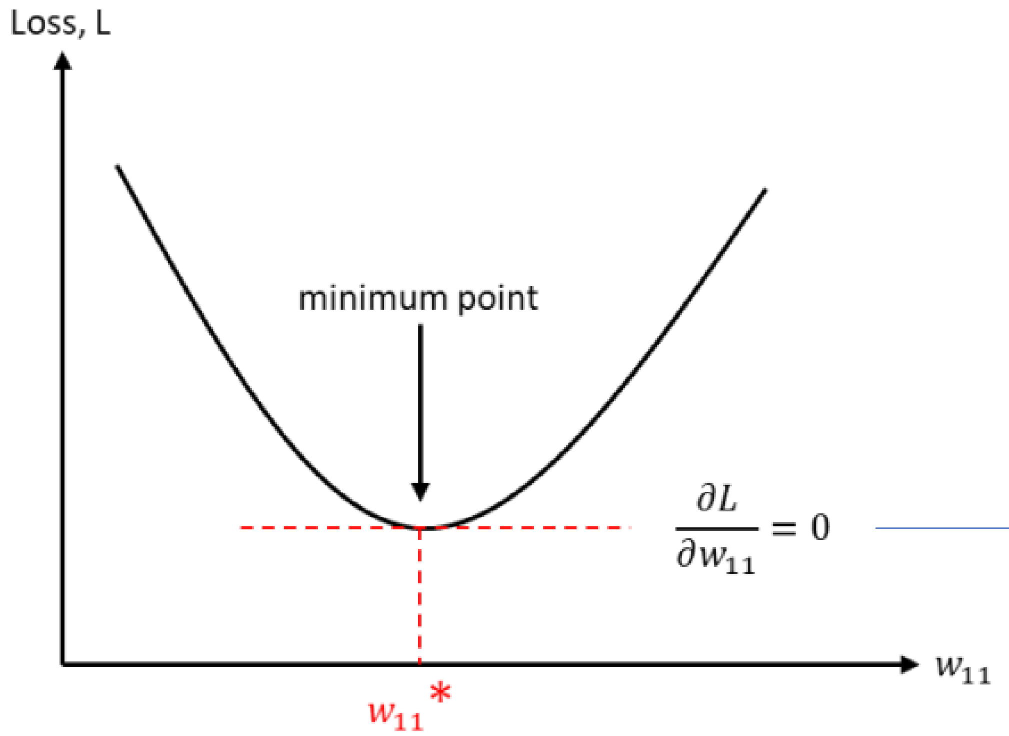


We have calculated the output!!!



Backward pass (backpropagation)

Loss function and its minimization by gradient descent

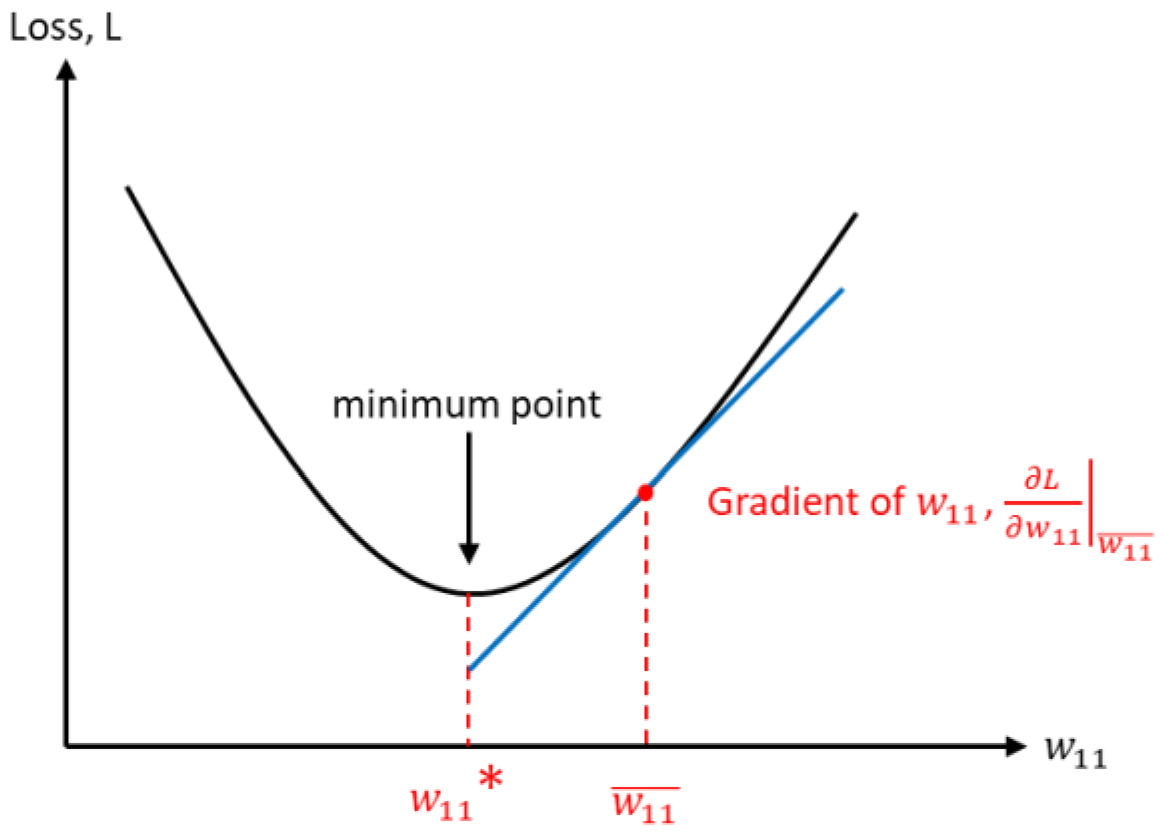


Too "expensive"

So, we resort to a different approach, that is the Gradient Descent (GD).

Backward pass (backpropagation)

Loss function and its minimization by gradient descent



Searching minimum point by gradient descent

$$w_{11}{}' = \overline{w_{11}} - \mu \left.\frac{\partial L}{\partial w_{11}}\right|_{\overline{w_{11}}} \tag{22}$$

where μ is a hyperparameter known as learning rate.

To express it in a familiar form you might read elsewhere, we rewrite Eqn. (22) below,

$$w_{11}{}' = w_{11} - \mu \frac{\partial L}{\partial w_{11}} \tag{23}$$

For the rest of weights and biases in our network, their update can be given as

$$w_{ji}{}' = w_{ji} - \mu \frac{\partial L}{\partial w_{ji}} \tag{24a}$$

$$u_j{}' = u_j - \mu \frac{\partial L}{\partial u_j} \tag{24b}$$

$$v_{ij}{}' = v_{ij} - \mu \frac{\partial L}{\partial v_{ij}} \tag{24c}$$

Gradients with respect to the parameters by chain-rule

For v_{ij}

$$v_{ij}' = v_{ij} - \mu \frac{\partial L}{\partial v_{ij}}$$

$$\frac{\partial L}{\partial v_{ij}} = 2\,(-E_1 \frac{\partial S_t}{\partial v_{ij}} - E_2 \frac{\partial T_t}{\partial v_{ij}}) \qquad (25a)$$

For u_j

$$u_j' = u_j - \mu \frac{\partial L}{\partial u_j}$$

$$\frac{\partial L}{\partial u_j} = 2\,(-E_1 \frac{\partial S_t}{\partial u_j} - E_2 \frac{\partial T_t}{\partial u_j}) \qquad (25b)$$

For w_{ji}

$$w_{ji}' = w_{ji} - \mu \frac{\partial L}{\partial w_{ji}}$$

$$\frac{\partial L}{\partial w_{ji}} = 2\,(-E_1 \frac{\partial S_t}{\partial w_{ji}} - E_2 \frac{\partial T_t}{\partial w_{ji}}) \qquad (25c)$$

Chain-rule

Now, we will discuss how to get the six gradients in Eqns. (24) which are, $\frac{\partial S_t}{\partial v_{ij}}$, $\frac{\partial T_t}{\partial v_{ij}}$, $\frac{\partial S_t}{\partial u_j}$, $\frac{\partial T_t}{\partial u_j}$, $\frac{\partial S_t}{\partial w_{ji}}$, and $\frac{\partial T_t}{\partial w_{ji}}$, by chain-rule. But, why chain-rule? For the following reasons:

i. it is effective.

ii. to prepare you for future discussion on auto-differentiation or autograd (remember these terms) which chain-rule is key.

i. Differentiation of z

$$z_j = w_{j1}x_k + w_{j2}y_k + u_j \tag{26}$$

thus,

$$\frac{\partial z_j}{\partial w_{j1}} = x_k \tag{27a}$$

$$\frac{\partial z_j}{\partial w_{j2}} = y_k \tag{27b}$$

$$\frac{\partial z_j}{\partial u_j} = 1 \tag{27c}$$

ii. Differentiation of sigmoid,

$$\sigma_j = \frac{1}{1 + e^{-z_j}} \tag{28}$$

thus,

$$\frac{\partial \sigma_j}{\partial z_j} = \sigma_j(1 - \sigma_j) \tag{29}$$

iii. Differentiation of the output (S_t and T_t) (Eqns. (13) and (14))

$$S_t = v_{1j}\sigma_j \tag{30a}$$

$$T_t = v_{2j}\sigma_j \tag{30b}$$

thus,

$$\frac{\partial S_t}{\partial \sigma_j} = v_{1j} \tag{31a}$$

$$\frac{\partial T_t}{\partial \sigma_j} = v_{2j} \tag{31b}$$

Now, we are ready to employ the chain-rule to obtain our output gradients with respect to the parameters (weights and biases).

i. Output gradient with respect to w_{ij}

$$\frac{\partial S_t}{\partial w_{j1}} = \frac{\partial S_t}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial z_j}\frac{\partial z_j}{\partial w_{j2}} \longrightarrow \frac{\partial S_t}{\partial w_{j1}} = v_{1j}x_k\sigma_j(1 - \sigma_j)$$

$$\frac{\partial S_t}{\partial w_{j2}} = \frac{\partial S_t}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial z_j}\frac{\partial z_j}{\partial w_{j2}} \longrightarrow \frac{\partial S_t}{\partial w_{j2}} = v_{1j}y_k\sigma_j(1 - \sigma_j)$$

Repeating the same process, the gradients for T_t can be given as,

$$\frac{\partial T_t}{\partial w_{j1}} = v_{2j}x_k\sigma_j(1 - \sigma_j)$$

$$\frac{\partial T_t}{\partial w_{j2}} = v_{2j}y_k\sigma_j(1 - \sigma_j)$$

ii. Output gradient with respect to u_j

$$\frac{\partial S_t}{\partial u_j} = \frac{\partial S_t}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial z_j}\frac{\partial z_j}{\partial u_j} \longrightarrow \frac{\partial S_t}{\partial u_j} = v_{1j}(1)\sigma_j(1-\sigma_j)$$

$$\frac{\partial T_t}{\partial u_j} = \frac{\partial T_t}{\partial \sigma_j}\frac{\partial \sigma_j}{\partial z_j}\frac{\partial z_j}{\partial u_j} \longrightarrow \frac{\partial T_t}{\partial u_j} = v_{2j}(1)\sigma_j(1-\sigma_j)$$

iii. Output gradient with respect to v_{ij}

$$\frac{\partial S_t}{\partial v_{1j}} = \sigma_j$$

$$\frac{\partial T_t}{\partial v_{2j}} = \sigma_j$$

Updating the network parameters (w_{ji}, u_j, v_{ij})

By inserting Eqns. (25) into (24) accordingly,

$$w'_{j1} = w_{j1} - 2\mu\,(-E_1 \frac{\partial S_t}{\partial w_{j1}} - E_2 \frac{\partial T_t}{\partial w_{j1}}) \tag{39a}$$

$$w'_{j2} = w_{j2} - 2\mu\,(-E_1 \frac{\partial S_t}{\partial w_{j2}} - E_2 \frac{\partial T_t}{\partial w_{j2}}) \tag{39b}$$

$$u'_j = u_j - 2\mu\,(-E_1 \frac{\partial S_t}{\partial u_j} - E_2 \frac{\partial T_t}{\partial u_j}) \tag{39c}$$

$$v'_{1j} = v_{1j} - 2\mu\,(-E_1 \frac{\partial S_t}{\partial v_{1j}}) \tag{39d}$$

$$v'_{2j} = v_{2j} - 2\mu\,(-E_2 \frac{\partial T_t}{\partial v_{2j}}) \tag{39e}$$

where,

E_1 and E_2 are given by Eqns. (17)

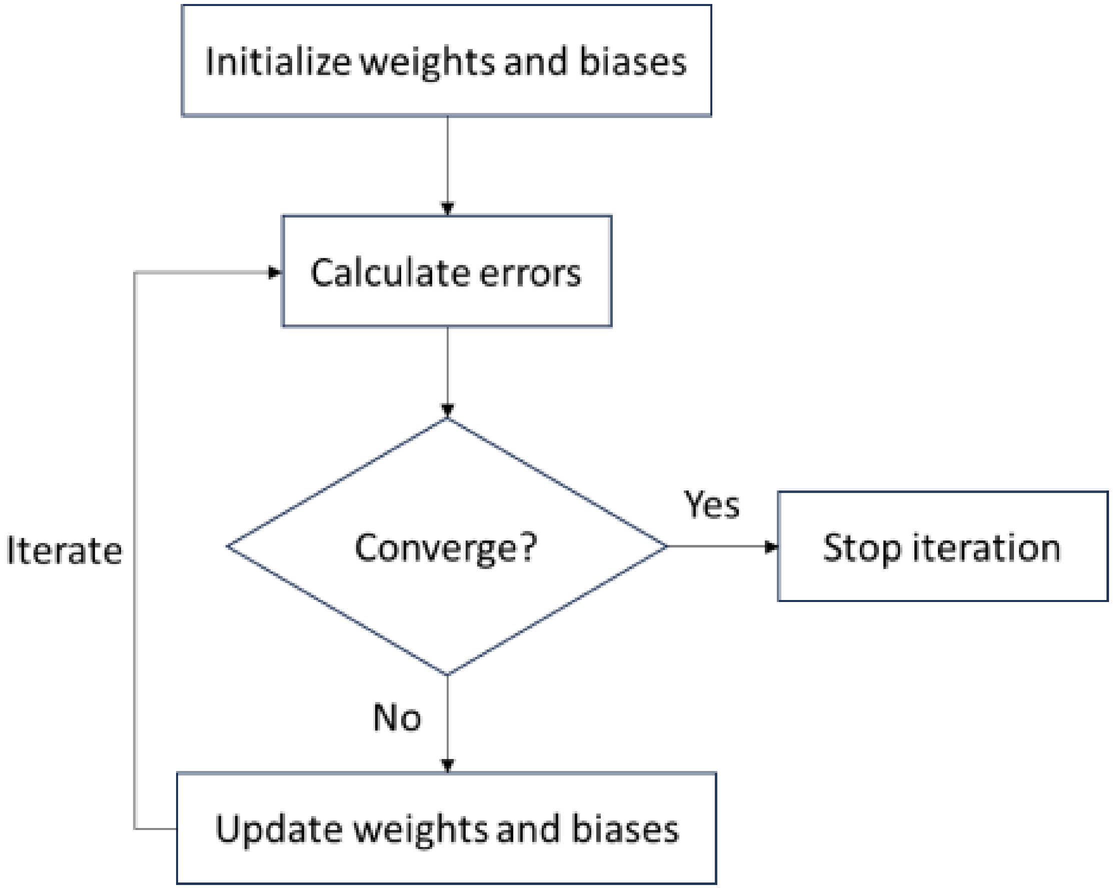
$\frac{\partial S_t}{\partial w_{j1}}$ and $\frac{\partial S_t}{\partial w_{j2}}$ are given by Eqns. (34)

$\frac{\partial T_t}{\partial w_{j1}}$ and $\frac{\partial T_t}{\partial w_{j2}}$ are given by Eqns. (35)

$\frac{\partial S_t}{\partial u_j}$ and $\frac{\partial T_t}{\partial u_j}$ are given by Eqns. (37)

$\frac{\partial S_t}{\partial v_{1j}}$ and $\frac{\partial T_t}{\partial v_{2j}}$ are given by Eqns. (38)

The network training flowchart



Flowchart of the training of network

Matlab code

The script

```
% STEP 1 : Generate inital Value for weight and biases
M = 2;
v = rand(2,M)-1/2;
u = rand(1,M)-1/2;
w = rand(2,M)-1/2;
```

The equations coded

$$[w] = \begin{bmatrix} w_{11} & w_{21} & \cdots & w_{m1} \\ w_{12} & w_{22} & \cdots & w_{m2} \end{bmatrix} \quad (2)$$

$$[v] = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1m} \\ v_{21} & v_{22} & \cdots & v_{2m} \end{bmatrix} \quad (3)$$

$$[u] = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix} \quad (4)$$

Box 1

The script

```
xk = (round(rand(1,N)*L,3));
yk = (round(rand(1,N)*L,3));
```

The equations coded

$$\{\bar{x}\} = \begin{Bmatrix} x_k \\ y_k \end{Bmatrix} = \begin{Bmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{Bmatrix} \qquad (2^*)$$

Box 2

The script

```
z = [];
for j = 1:M %loop over neuron
    zj = w(1,j)*xk + w(2,j)*yk + u(j);
    z = cat(1,z,zj);
```

The equations coded

$$z_m = w_{m1}x_k + w_{m2}y_k + u_m \tag{10}$$

Box 3

The script

```
sig = 1 ./ (1 + exp(-z));
dsigdz = sig .* (1-sig);
```

The equations coded

$$\sigma_m = \frac{1}{1 + e^{-z_m}} \qquad \text{(12a) or (28)}$$

$$\frac{\partial \sigma_j}{\partial z_j} = \sigma_j(1 - \sigma_j) \tag{29}$$

Box 4

The script

```
St = 0;
Tt = 0;
for j = 1:M %loop over neuron
    St = St + v(1,j) *sig(j,:);
    Tt = Tt + v(2,j) *sig(j,:);
end
```

The equations coded

$$S_t = v_{1j}\sigma_j \quad (30a)$$

$$T_t = v_{2j}\sigma_j \quad (30b)$$

Box 5

The script

```
% STEP 5 : Calculate the actual equations
s = c1*xk + c2*yk;
t = d1*xk + d2*yk;
```

The equations coded

$$3x + 2y = s \quad (1a)$$

$$2x + 6y = t \quad (1b)$$

Box 6

The script

```
% STEP 6 : Calculate the residual errors
E1 = (s - St);
E2 = (t - Tt);
```

The equations coded

$$E_1 = (s - S_t) \quad (17a)$$

$$E_2 = (t - T_t) \quad (17b)$$

Box 7

The script

```
% diff over v1
dStdv1 = -sig(j,:);

% diff over v2
dTtdv2 = -sig(j,:);
```

The equations coded

$$\frac{\partial S_t}{\partial v_{1j}} = \sigma_j \quad (38a)$$

$$\frac{\partial T_t}{\partial v_{2j}} = \sigma_j \quad (38b)$$

Box 8

The script

```
% diff over u
dStdu = -v(1,j) *dsigdz(j,:);
dTtdu = -v(2,j) *dsigdz(j,:);
```

The equations coded

$$\frac{\partial S_t}{\partial u_j} = v_{1j}(1)\sigma_j(1-\sigma_j) \qquad (37a)$$

$$\frac{\partial T_t}{\partial u_j} = v_{2j}(1)\sigma_j(1-\sigma_j) \qquad (37b)$$

Box 9

The script

```
% diff over w1
dStdw1 = -v(1,j) *xk .*dsigdz(j,:);
dTtdw1 = -v(2,j) *xk .*dsigdz(j,:);

% diff over w2
dStdw2 = -v(1,j) *yk .*dsigdz(j,:);
dTtdw2 = -v(2,j) *yk .*dsigdz(j,:);
```

The equations coded

$$\frac{\partial S_t}{\partial w_{j1}} = v_{1j}x_k\sigma_j(1-\sigma_j) \qquad (34a)$$

$$\frac{\partial S_t}{\partial w_{j2}} = v_{1j}y_k\sigma_j(1-\sigma_j) \qquad (34b)$$

$$\frac{\partial T_t}{\partial w_{j1}} = v_{2j}x_k\sigma_j(1-\sigma_j) \qquad (35a)$$

$$\frac{\partial T_t}{\partial w_{j2}} = v_{2j}y_k\sigma_j(1-\sigma_j) \qquad (35b)$$

Box 10

The script

```
% STEP 8 : Update the weight and biases
v(1,j) = v(1,j) - eta*sum(2*E1.*dStdv1);
v(2,j) = v(2,j) - eta*sum(2*E2.*dTtdv2);
u(j)   = u(j)   - eta*sum(2*E1.*dStdu  + 2*E2.*dTtdu );
w(1,j) = w(1,j) - eta*sum(2*E1.*dStdw1 + 2*E2.*dTtdw1);
w(2,j) = w(2,j) - eta*sum(2*E1.*dStdw2 + 2*E2.*dTtdw2);
```

The equations coded

$$w'_{j1} = w_{j1} - 2\mu\,(-E_1\frac{\partial S_t}{\partial w_{j1}} - E_2\frac{\partial T_t}{\partial w_{j1}}) \qquad (39a)$$

$$w'_{j2} = w_{j2} - 2\mu\,(-E_1\frac{\partial S_t}{\partial w_{j2}} - E_2\frac{\partial T_t}{\partial w_{j2}}) \qquad (39b)$$

$$u'_j = u_j - 2\mu\,(-E_1\frac{\partial S_t}{\partial u_j} - E_2\frac{\partial T_t}{\partial u_j}) \qquad (39c)$$

$$v'_{1j} = v_{1j} - 2\mu\,(-E_1\frac{\partial S_t}{\partial v_{1j}}) \qquad (39d)$$

$$v'_{2j} = v_{2j} - 2\mu\,(-E_2\frac{\partial T_t}{\partial v_{2j}}) \qquad (39e)$$

Box 11

That’s it!

References:

[1] Zhuhadar, Lily Popova & Lytras, Miltiadis. (2023). The Application of AutoML Techniques in Diabetes Diagnosis: Current Approaches, Performance, and Future Directions. Sustainability. 15. 13484. 10.3390/su151813484

[2] Sanubari, A. R., Kusuma, P. D., Setianingsih, C., Flood Modelling and Prediction Using Artificial Neural Network, The 2018 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)

[3] Michael Nielsen, Neural Networks and Deep Learning, http://neuralnetworksanddeeplearning.com

[4] Eva Prašnikar, Martin Ljubič. Andrej Perdih, Jure Borišek, Machine learning heralding a new development phase in molecular dynamics simulations, Artifcial Intelligence Review (2024) 57:102

[5] Kumar Puran Tripathy, Ashok K Mishra, Deep Learning in Hydrology and Water Resources: concepts, methods, applications and research directions, Journal of Hydrology, 628 (2024)