Predicting Drought Indices in Nakhon Ratchasima Province using a Deep Belief Network with Restricted Boltzmann Machines

Sureeluk Ma

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Applied Mathematics
Prince of Songkla University
2018

Thesis Title	Predicting Drought Indices in Nakhon Ratchasima Province using a Deep Belief Network with Restricted Boltzmann-Machines
Author	Miss Sureeluk Ma
Major Program	Applied Mathematics

Major Advisor

.....

(Dr. Tatdow Pansombut)

Co-Advisor

.....

(Dr. Pakwan Riyapan)

Examining Committee :

..... Chairperson

(Asst. Prof.Dr. Nifatamah Makaje)

.....

(Dr. Tatdow Pansombut)

.....

(Dr. Pakwan Riyapan)

.....

(Asst. Prof.Dr. Arthit Intarasit)

.....

(Asst. Prof.Dr. Usa Sammapun)

The Graduate School, Prince of Songkla University has approved this thesis as partial fulfillment of the requirements for the Master of Science Degree in Applied Mathematics.

.....

(Prof.Dr. Damrongsak Faroongsarng)

Dean of Graduate School

This is to certify that the work here submitted is the result of the candidate's own investigations. Due acknowledgement has been made of any assistance received.

..... Signature

(Dr. Tatdow Pansombut)

Major Advisor

..... Signature

(Dr. Pakwan Riyapan)

Co-Advisor

..... Signature

(Miss Sureeluk Ma)

Candidate

I hereby certify that this work has not been accepted in substance for any degree, and is not being currently submitted in candidature for any degree.

..... Signature

(Miss Sureeluk Ma)

Candidate

ชื่อวิทยานิพนธ์ การทำนายดัชนีความแห้งแล้งในจังหวัดนครราชสีมาโดยใช้ Deep Belief Network กับ Restricted Boltzmann Machines

ผู้เขียน นางสาวสุรีลักษณ์ มะ

สาขาวิชา คณิตศาสตร์ประยุกต์

ปีการศึกษา 2560

บทคัดย่อ

ในการศึกษาครั้งนี้ เราจะตรวจสอบความสามารถของ deep learning ในการทำนายโดยใช้ข้อมูลชุดเวลา ซึ่งเป็นข้อมูลน้ำฝนจากจังหวัดนครราชสีมา ภาคตะวันออกเฉียงเหนือของประเทศไทย ข้อมูลน้ำฝนจะถูกแปลงเป็นดัชนีของ standardized precipitation index (SPI) ในหลายๆ ช่วงเวลา ข้อมูล SPI ในแต่ละช่วงเวลาจะถูกเรียนรู้โดย deep belief network ซึ่งเป็นโครงสร้างที่ประกอบจากหลายๆ restricted Boltzmann machines ประกอบด้วยกัน ตัวแปรแต่ละตัวในวิธีการนี้จะถูกเรียนรู้ผ่านกระบวนการ 2 ขั้นตอน กระบวนการแรกคือ unsupervised path ซึ่งเรียนรู้โดยใช้อัลกอริทึม minimized contrastive divergent กระบวนการที่สองคือ supervised path ซึ่งเรียนรู้โดยใช้อัลกอริทึม backpropagation จากการเปรียบเทียบความแม่นยำของค่าดัชนี SPI ของแต่ละช่วงเวลา พบว่า SPI ที่ช่วงเวลา 12 เดือน มีค่าความแม่นยำสูงกว่าช่วงเวลา 3, 6 และ 9 เดือน

Thesis Title	Predicting Drought Indices in Nakhon Ratchasima Province using a Deep Belief Network with Restricted Boltzmann-Machines
Author	Miss Sureeluk Ma
Major Program	Applied Mathematics

ABSTRACT

In this study, we examine the ability of deep learning in making prediction from time series data. First, the precipitation data from Nakhon Ratchasima province in northeastern region of Thailand is converted into various types of standardized precipitation index (SPI). Next, for each SPI, a deep belief network, consisting of restricted Boltzmann machines, learns its parameters from data through unsupervised path using minimized contrastive divergent algorithm follow by supervised path using backpropagation algorithm. Last, the prediction accuracies from all types of the standardized precipitation index are evaluated and compared. The result shows that the long term SPI of 12 months makes more accurate prediction than the short term SPI of 3, 6, and 9 months.

Acknowledgement

First and foremost, I would like to thank the Almighty God for seeing me through this study. I would like to show my profound gratitude to my advisors, Dr. Tatdow Pansombut and Dr. Pakwan Riyapan for their guidance, advice, encouragement and motivation for my study. Their efforts are deeply appreciated and I am happy to have them through this study.

I would like to also thank the Applied Mathematics program committee chaired by Asst. Prof.Dr. Nifatamah Makaje for giving me the opportunity to enhance my knowledge through this program. I would like to appreciate the Department of Mathematics and Computer Science headed by Asst. Prof.Dr. Areeyuth Sama-Ae for providing me with a conducive atmosphere to do my study.

I would like to acknowledge the Centre of Excellence in Mathematics, Commission on Higher Education (CHE) for funding my studies. I am also grateful to the Faculty of Science and Technology and the Office of the Graduate School, Pattani campus.

I appreciate my colleagues and friends in Applied Mathematics as well as the Research Methodology Program for being there when I needed them most.

Finally, I would like to thank my parents for their help and support throughout my entire educational life will never be forgotten. God richly bless them.

Sureeluk Ma

Contents

List of Figures

List of Tables

Chapter1

INTRODUCTION

1.1 Overview

The attention of the international community and other organizations have been drawn to the problem of environmental degradation and its effect on water sources. A major cause of such degradation is drought and its devastating effects have attracted the attention of various government and Non-Governmental Organizations (NGOs). Unfortunately, drought is also known to be the least understood natural disaster (Chen *et al.*, 2012). Droughts continue to occur and cause serious problems to human survival and food production in most parts of the world. For instance, in 2009 and 2010, severe droughts raked through Southwest China, causing a lot of economic damage and resulting in significant number of deaths and has consistently been ranked as the first in all natural disasters the populous Asian country (Chen *et al.*, 2012).

For the past half a decade, severe droughts have caused so much damage to agricultural activities in most provinces in Thailand. These droughts were mainly caused by the drop in annual rainfall below a 30-year average (SCB EIC, 2016). Looking at the devastating effects of droughts in Thailand, it is important to

model and predict drought condition in order to know when and where droughts are likely to occur. Drought prediction has become an important research that focuses on planning and managing water resources. Such predicting methods are based on past rainfall data.

Generally, data on rainfall are considered as time series data. Time series refer to the measurement of a phenomenon over a specific period. Time Series data is the set of observations that are obtained from such measurements. Analyzing time series data involves different processes which includes modeling a particular phenomenon, describing the behavior of the data, and evaluating factors that may be associated with the behavior (Hrasko *et al.*, 2015). Time series forecasting can be described as the process of making future predictions based on time series data. For over half a century, a lot of studies have been conducted on time series forecasting. These studies have resulted in the development of models that are similar to Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA) (Box *et al.*, 1976). For the most part of the 20th Century, times series forecasting was mainly modeled as linear problems by means of regression method, however, since the late 1990s, machine learning has gained popularity in nonlinear estimations. These methods which include artificial neural network have been proven to have much predictive power and applicable to various real-life problems (Hrasko *et al.*, 2015).

Artificial neural network has extensive adaptability and the ability to learn non-linear problems (Shen *et al.*, 2013). There are two interesting features in artificial neural network. Firstly, artificial neural network has general nonlinear

function which can approximate any continuous function with desired accuracy (George, 1989). Therefore, artificial neural network is capable for solving several complex problems. Secondly, artificial neural network is a nonparametric data-driven model and it does not need the restrictive assumption on the process of the data generation. As indicated by Crone and Nikolopoulos (2007), there are more than 5000 publications on artificial neural network as a tool for forecasting. The predictor for these applications can be listed as multi-layer perceptron (MLP), recurrent neural networks (RNN), radial basis function networks (RBFN), and several other varieties.

Even though several applications of artificial neural network are successful in forecasting .There are still lacking when artificial neural network apply to real problems. The first problem is how to decide the structure of artificial neural network. The second problem is how to decide the initial weight values in artificial neural network. The third problem is how to find suitable learning rate during the learning. For these problems, different researchers have provided solutions as follows.

Hecht (1992) proved the first problem that multilayer-layer perceptron with one hidden layer and enough neurons in hidden layers can realize approximation of nonlinear function. For the second problem, the suitable initial weight values may accelerate the learning convergence. Therefore, Hinton *et al.* (2006) proposed a deep belief network (DBN) which the learning consists of two paths. The first path is unsupervised path in which the learning of the model does not need feedback (no label). The weight values obtained from in this path will be used to initialize

the weight values in the second path, which is the supervised path. For the third problem, high learning rate might destabilize the learning convergence. Too small learning rate might be very time consuming to converge. Solving of these three problems in artificial neural network by difference researchers, we decided to use deep belief network to predict drought.

Deep belief networks (DBNs) are probabilistic generative models that are made from restricted Boltzmann machines (RBMs) stacking together. This deep belief network has shown success in its application to many real time problems (Hinton *et al.*, 2006). Since there is no mature method to determine network structure of deep belief network, therefore, experimental method will be used in this research. Only one best network structure with the smallest root mean square error will be used to predict drought in this research.

1.2 Objectives

The objectives in this research are

1. To study the scientific measurement of drought condition by exploring how drought indices (indicators of drought) are calculated from data.

2. To implement a deep belief network with restricted Boltzmann machine using the precipitation time series data from Nakhon Ratchasima province, Thailand.

3. To apply the implemented deep belief network with restricted Boltzmann machine to make predictions about the drought condition in Nakhon Ratchasima province, Thailand.

Chapter2

LITERATURE REVIEW

The importance of forecasting drought has resulted in lots of studies in different settings to explain this natural phenomenon.

In 1997, Lohani and Loganathan conducted a study with the main objective to predict early warning signs of droughts and to propose drought management decisions. They used nonhomogeneous Markov chains with the Palmer Drought Severity Index (PDSI) to describe and characterize drought behavior. With applications to data from the climatic division of Virginia, they were able to propose an early-warning system that gives first hand update on possible droughts and their severity. The system that they proposed was a form of a decision tree and was effective in drought management (Lohani *et al.*, 1997).

Han *et al.* (2010) used remote sensing data to forecast drought in Guanzhong, China. The study used ARIMA models with Vegetation Temperature Condition Index (VTCI) series to predict future possibilities of drought. The results from the study indicated that the ARIMA model had better forecasting accuracy with regards to the remote sensing data. There are many models in ARIMA. In this study, they concluded that ARIMA(1,0,0) or AR(1) process developed for VTCI

can be used to forecast drought in Guanzhong Plains. However, they failed to apply ARIMA models to VTCI series using real time series data set (Han *et al.*, 2010).

In 2012, Chen *et al.* proposed a deep belief network and back propagation (BP) for short-term drought index prediction. They used data from four hydrologic stations in China to calculate different time scales of standardized precipitation index (SPI) and predicted drought by using time series data of monthly rainfall from January 1958 to 2006. The stations were Begbu, Fuyang, Xuchang, and Zhumadian in Huaihe River Basin in the eastern part of China. Their results showed that the deep belief network has a higher accuracy in drought prediction based on SPI than back propagation and the error result showed that deep belief network model was suitable to predict drought in the Huaige River Basin (Chen *et al.*, 2012).

Chapter3

METHODOLOGY

This chapter represents the methodology that would be used in this research. The aim in this research is to predict drought by using a deep belief network with restricted Boltzmann machines. The data used to predict drought is monthly rainfall data, which is a kind of time series data. Therefore, Section 3.1 will be explained time series data. To predict drought, we have to transform rainfall data into drought index. There are several methods to transform, where some important methods are explained in Section 3.2. The main factor that causes drought is the lack of rain, hence, this study will focus on standardized precipitation index (SPI), the detail of this method will be explained in Section 3.3. Since we want to formulate the energy function of restricted Boltzmann machine and continuous restricted Boltzmann machine using the energy function of Hopfield network, the detail about Hopfield network will be explained in Section 3.4. Section 3.5 is explained the detail about restricted Boltzmann machine including of minimizing contrastive divergence algorithm for restricted Boltzmann machine. Section 3.6 is explained the detail about continuous restricted Boltzmann machine including of minimizing contrastive divergence algorithm for continuous restricted

Boltzmann machine, which this method will be used in this research. Section 3.7 is explained the detail of feedforward neuron network with back-propagation algorithm. Deep belief network is stacked of restricted Boltzmann machine, which the detail is explained in Section 3.8. Learning a deep belief network consists of two paths, unsupervised path and supervised path, which the detail is explained in Section 3.9.

3.1 Time series data

Time series analysis has become increasingly important in various fields of research, such as business, economics, engineering, and medicine. The analysis of time series data involves the study in the following aspects (Kumar, 2008):

i **Past behavior:**

Time series analysis can be used to study the past behavior of businesses. It reveals the trends in sales and helps in business investments.

ii **Forecasting:**

Forecasting is defined as the process for predicting future outcomes based on past events. By using time series, the history of these events can help to decide what happens in the future.

iii **Evaluating achievements:**

Time series is a tool that can help to evaluate achievements. If there is a good performance in a business, a time series analysis can show an upward trend in profits. A downward trend will prompt management of the business to make

new policies.

Time series analysis is an attempt to model phenomenon and to describe the behavior of many real problem domains such as financial markets, signal processing, weather forecasting and others. The problems in these domains are generally complex and cannot be easily solved. They require advanced techniques.

As explained by Yaffee *et al.* (2000), rainfall data can be considered a time series data. Time series data is a set of observations that is recorded over a specific time. Time series data may be measured continuously or discretely. Daily closing stock price is an example of time series data that is measured continuously. Most data used in social sciences are measured at regular intervals. These kinds of times series data are discrete (Yaffee *et al.*, 2000). Time series data occur in variety of fields such as in business, economics, and engineering. Economists always observe daily closing stock prices, weekly interest rates, monthly price, and yearly earnings. In engineering, sound, electric signals, and rainfall are always observed.

Box *et al.* (1976) explained that time series data contains one or more of the following properties: trends, seasonality, regular, and irregular patterns (Box a*et al.*, 1976).

Trend

A trend exists when the data have a long-term increase or decrease (Box *et al.*, 1976). Graph in Figure 3.1 shows the data of Quarterly Gross Domestic Product (Australia's National Statistical Agency) which x-axis represents year and y-axis represents million cent. There is an obvious upward trend over time in this graph.

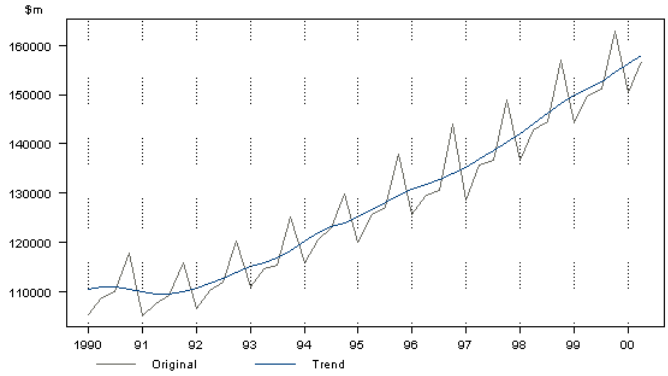

Figure 3.1: The Quarterly Gross Domestic Product which x-axis represents year and y-axis represents million cent

Source: Hyndman *et al.* (2018)

Seasonal patterns

A seasonal pattern exists when the data are influenced by seasonal factors such as the quarter of the year, month, or day and these seasons are always fixed (Box *et al.*, 1976). For instance, the monthly retails sales in New South Wales (NSW) Retail Department Stores (Australia's national statistical agency) is shown in Figure 3.2. In this figure, x-axis represents year and y-axis represents million cent. Seasonality in this graph has consistent direction and approximately the same magnitude every year. The graph depicts a strongly seasonal series and there is a large seasonal increase over time as the trend in the figure.

Cyclic patterns

A time series data is said to depict cyclic patterns when it exhibits a rises and falls that are not fixed to a particular period (Box *et al.*, 1976). An example of data that shows cyclic behavior is the monthly housing sales (Australia's national statistical agency) shown in Figure 3.3. The x-axis represents year and y-axis represents million cent. This graph has consistent and approximately the

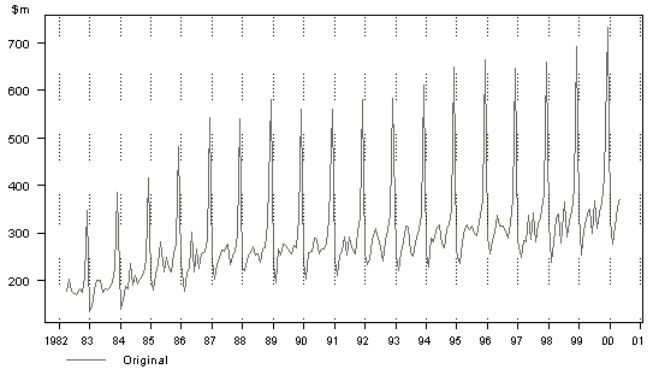

Figure 3.2: The Monthly Retail Sales in New South Wales Retail Department Stores which x-axis represents year and y-axis represents million cent

Source: Hyndman *et al.* (2018)

same magnitude with not fixed particular period.

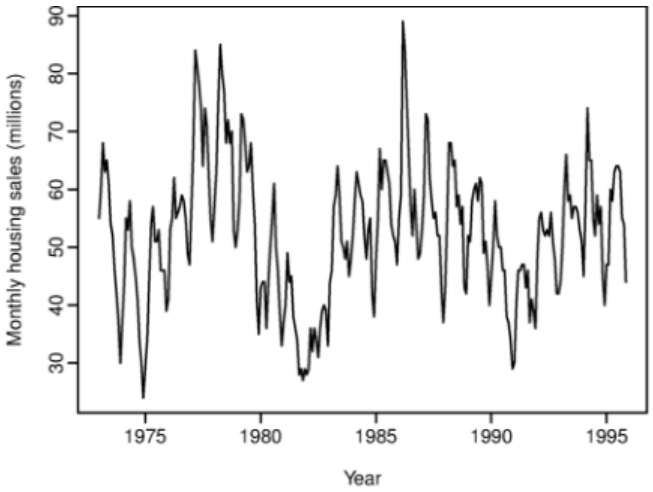

Figure 3.3: The Monthly Value of Building Approvals, Australian Capital Territory (ACT) which x-axis represents year and y-axis represents million cent

Source: Hyndman *et al.* (2018)

Irregular patterns

An irregular pattern is defined as the remains of a trend, seasonal patterns, and cyclic patterns (Box *et al.*, 1976). The plot of monthly value for building approvals, Australian Capital l Territory (ACT) in Figure 3.4 shows high irregular patterns. The x-axis represents year and y-axis represents million cent

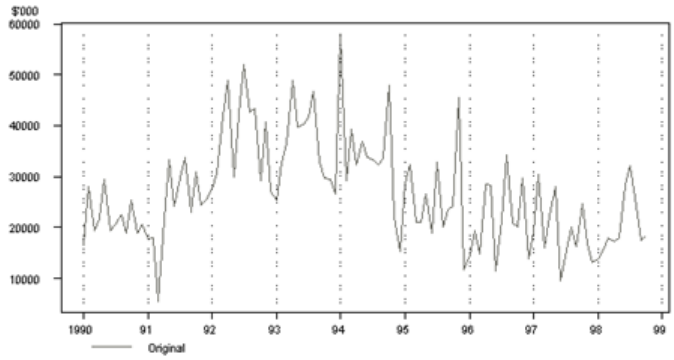

Figure 3.4: The Monthly Value of Building Approvals, Australian Capital Territory (ACT) which x-axis represents year and y-axis represents million cent

Source: Hyndman *et al.* (2018)

3.2 Drought index

After verifying the properties of the rainfall data, drought indices would be calculated from assimilating drought indicators into a single numerical value. A drought index gives an in-depth information for drought analysis and this is more usable in comparison with actual data from indicators (Hayes, 2006). The main factor that causes drought in Thailand is inadequate rain, hence, this study will focus on rainfall. There are many methods to transform raw data into drought index, some of these methods will be explained as follows:

3.2.1 Effective Drought Index

Effective drought index (EDI) was discovered by Byun and Wilhite in 1999, which uses daily precipitation data to compute. The goodness of EDI is required for a single input, it is daily precipitation data. Therefore, EDI is possible to compute for any location (Byun *et al.*, 1999). The weakness is recording of daily precipitation data may not be possible. Therefore, it will have a problem to

compute EDI (Byun *et al.*, 1999).

The index value of EDI ranges from less than -2 to greater than 2, (-2, 2). The index of less than or equal to -2 stands for extremely dry. The index of greater than or equal to 2 stands for extremely wet (Byun *et al.*, 1999). Byun and Wilhite have been defined criteria to evaluate drought index which is shown in Table 3.1.

Table 3.1: Degree of violence of effective drought index (EDI)

EDI value	**Degree of violence**
2+	Extremely wet conditions
0.99 to -0.99	Near normal conditions
-1 to -1.49	Moderate drought
-1.5 to -1.99	Severe drought
-2.00 and less	Extremely dry conditions

3.2.2 Generalized Monsoon Index

Generalized monsoon index (GMI) was developed by Achutuni *et al.* in 1982. It is agro-meteorological index. GMI was calculated by using monthly rainfall during the monsoon season (Achutuni *et al.*, 1982). This method is used by the meteorological department of Thailand to evaluate the effect of rainfall on agriculture. The goodness of this method is to see the effect on growing plant due to lack of moisture. The weakness is GMI index considers the rainfall data during the monsoon season. Therefore, it is not possible to evaluate other effect on agriculture.

Table 3.2 shows the degree of violence of generalized monsoon index which the possible values can be 0 to 100. The index of too small value means that drought impact and possible crop failure. The index of too high value means that

possible excessive moisture. GMI_{pct} in this table is generalized monsoon index in percentile rank.

Table 3.2: Degree of violence of generalized monsoon index in percentile rank (GMI_{pct})

GMI_{pct} **value**	**Degree of violence**
100 to 91	Possible excessive moisture
90 to 61	Possible above normal crop
60 to 41	Normal crop
40 to 31	Moderate drought impact on crop
30 to 21	Drought impact on crop
20 to 0	Severe drought impact and possible crop failure

3.2.3 Moisture Available Index

In 1975, George determined moisture index on moisture available index (MAI) that was useful for plants. This index can be calculated from the ratio obtained from dependable rainfall and evapotranspiration of the probability of monthly rainfall at 75%. The goodness of this index is consideration of moisture which influences crop production. Therefore, this index is useful for the place that lack of moisture. There are many factors of crop production. So the weakness of this index is consideration only moisture. To evaluate the moisture Hargreaves has determined the index of degree of violence which is shown in Table 3.3. If the index value is close to 0, then this means that plants lacks water however if the index value is greater than 1.33, then the plants get too much water (George, 1975).

Table 3.3: Degree of violence of moisture available index (MAI)

MAI value	**Degree of violence**
1.34+	Plants get too much water
1.33 to 1.01	Plants get enough water
0.68 to 1.00	Plants lack little water
0.34 to 0.67	Plants lack moderate water
0.00 to 0.33	Plants lack severe water

3.2.4 Standardized Precipitation Index

The standardized precipitation index (SPI) was developed by Mckee *et al.* (1993) of the Colorado Climate Center. It is a tool developed to monitor drought and it is based on the probability of precipitation at any given time, using historical rainfall (Mckee *et al.*, 1993). The goodness of SPI required only one input monthly rainfall, it is not too difficult to compute. Furthermore, SPI can be computed for difference time-scale, it provides for short and long term of drought. The weakness, there are many factors cause drought. But SPI can account for only the lack of rain.

Table 3.4 defines the violence of drought by Mckee *et al.* The index value ranges from less than -2 to greater than 2, (-2, 2). The index value of less than or equal to -2 means that extremely dry. The index value of greater than or equal to 2 means that extremely wet.

The standardized precipitation index has been used by many drought planners such as research centers, universities and meteorological and hydrological centers across the globe to monitor drought and early warning signals (Svoboda and Wood, 2012). In Nakhon Ratchasima province, the main factor that caused drought is lack of rain, hence, this study will focus on standardized precipitation

index. This method will be explained in Section 3.3.

Table 3.4: Degree of violence of standardized precipitation index (SPI)

SPI value	**Degree of violence**
2.0+	Extremely wet
1.5 to 1.99	Very wet
1.0 to 1.49	Moderately wet
-0.99 to 0.99	Near normal
-1.0 to -1.49	Moderately dry
-1.5 to -1.99	Severely dry
-2 and less	Extremely dry

3.3 Standardized Precipitation Index

The standardized precipitation index (SPI) was developed by Mckee *et al.* (1993) of the Colorado Climate Center. It is a tool developed to monitor drought and based on the probability of precipitation at any given time, using historical rainfall (Mckee *et al.*, 1993).

The gamma probability distribution has proven very effective in providing a good fit for precipitation distribution (Thom, 1958), thus, it is used to estimate the standardized precipitation index. The gamma distribution is based on the gamma function given by

$$\Gamma(\gamma) = \int_0^\infty x^{\gamma-1} e^{-x} dx,$$

where

γ is a shape parameter which $\gamma > 0$,

General gamma distribution

Suppose that a random variable x has gamma distribution with the shape parameter γ and scale parameter β, x has probability density function f which is given

$$f(x) = \frac{1}{\beta^{\gamma}\Gamma(\gamma)} x^{\gamma-1} e^{\frac{-x}{\beta}},$$

where β is scale parameter and $\beta > 0$.

Computation of SPI involves fitting a gamma probability density function. Therefore, parameters β and γ must be estimated. Maximum likelihood is a tool for estimating the parameters given observations. This method can be applied to a great variety of statistical problems by finding the parameter values that maximize the likelihood of observations. Therefore, maximum likelihood method will be used to optimally estimate β and γ.

Maximum likelihood is a method of estimating parameters given observations. This method attempts to find the parameters that maximize the likelihood function. The likelihood function is a function of parameters given observations, which the likelihood function is equivalent to the probability of observations given parameters. Firstly, the likelihood function is constructed by

$$L(\gamma, \beta|x) = p(x|\gamma, \beta).$$

Suppose that we have n independent observations, they are $x_1, x_2, x_3, ..., x_n$. Thus the likelihood function is given by

$$\begin{aligned} L(\gamma, \beta|x) &= \prod_{i=1}^{n} f(x_i|\gamma, \beta) \\ &= \left(\frac{1}{\beta^\gamma \Gamma(\gamma)} x_1^{\gamma-1} e^{\frac{-x_1}{\beta}}\right)\left(\frac{1}{\beta^\gamma \Gamma(\gamma)} x_2^{\gamma-1} e^{\frac{-x_2}{\beta}}\right)\cdots\left(\frac{1}{\beta^\gamma \Gamma(\gamma)} x_n^{\gamma-1} e^{\frac{-x_n}{\beta}}\right) \\ &= \left(\frac{1}{\beta^\gamma \Gamma(\gamma)}\right)^n (x_1 x_2 \cdots x_n)^{\gamma-1} e^{-\frac{(x_1+x_2+...+x_n)}{\beta}}. \end{aligned}$$

Computing the likelihood function might be difficult. Since logarithm is monotonic function (Qi, 2007) which can be achieved its maximum value at the same point as the function itself, therefore, the log-likelihood function $ln\left(L(\gamma, \beta|x)\right)$ will be used.

$$\begin{aligned} ln\left(L(\gamma, \beta|x)\right) &= ln\left(\left(\frac{1}{\beta^\gamma \Gamma(\gamma)}\right)^n (x_1 x_2 \cdots x_n)^{\gamma-1} e^{-\frac{(x_1+x_2+...+x_n)}{\beta}}\right) \\ &= -nln\left(\beta^\gamma \Gamma(\gamma)\right) + (\gamma-1)ln(x_1 x_2 \cdots x_n) - \frac{1}{\beta}(x_1 + x_2 + ... + x_n) \\ &= -n\left(\gamma ln\beta + ln\Gamma(\gamma)\right) + (\gamma-1)ln(x_1 x_2 \cdots x_n) - \frac{1}{\beta}(x_1 + x_2 + ... + x_n). \end{aligned}$$

To find the maximum of a function, the derivative where the slope of tangent line is zero is required. Assume that the first derivative of the log-likelihood function is zero at $\beta = \hat{\beta}$ and $\gamma = \hat{\gamma}$, thus we get,

$$\frac{\partial}{\partial \beta}\left(lnL(\hat{\gamma}, \hat{\beta}|x)\right) = \frac{-n\hat{\gamma}}{\hat{\beta}} + \frac{1}{\hat{\beta}^2}\sum_{i=1}^{n} x_i = 0, \tag{3.1}$$

$$\frac{\partial}{\partial \gamma}\left(lnL(\hat{\gamma}, \hat{\beta}|x)\right) = -nln\hat{\beta} - n\frac{\partial}{\partial \gamma} ln\Gamma(\hat{\gamma}) + \sum_{i=1}^{n} lnx_i = 0. \tag{3.2}$$

From Equation (3.1),

$$\frac{-n\hat{\gamma}}{\hat{\beta}} + \frac{1}{\hat{\beta}^2}\sum_{i=1}^{n} x_i = 0$$
$$\frac{1}{\hat{\beta}^2}\sum_{i=1}^{n} x_i = \frac{n\hat{\gamma}}{\hat{\beta}}.$$

Hence, we can compute $\hat{\beta}$ as Equation (3.3)

$$\hat{\beta} = \frac{\bar{x}}{\hat{\gamma}}, \tag{3.3}$$

where

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}.$$

Multiplying Equation (3.2) by $\frac{1}{n}$, we obtain,

$$-ln\hat{\beta} - \frac{\partial}{\partial\hat{\gamma}} ln\Gamma(\hat{\gamma}) + \frac{1}{n}\sum_{i=1}^{n} lnx_i = 0. \tag{3.4}$$

The first order derivative of the logarithm of the gamma function $\left(\frac{\partial}{\partial\hat{\gamma}} ln\Gamma(\hat{\gamma})\right)$ is referred to digamma function $\psi(\hat{\gamma})$. In a simplified form, Equation (3.4) becomes

$$-ln\hat{\beta} - \psi(\hat{\gamma}) + \frac{1}{n}\sum_{i=1}^{n} lnx_i = 0. \tag{3.5}$$

To get $ln\hat{\beta}$ as in Equation (3.5), we will take natural logarithm in Equation (3.3)

$$\begin{aligned} ln\hat{\beta} &= ln\frac{\bar{x}}{\hat{\gamma}} \\ &= ln\bar{x} - ln\hat{\gamma}. \end{aligned}$$

Substituting $ln\hat{\beta}$ into Equation (3.5), we obtain,

$$\begin{aligned} -(ln\bar{x} - ln\hat{\gamma}) - \psi(\hat{\gamma}) + \frac{1}{n}\sum_{i=1}^{n} lnx_i = 0 \\ ln\hat{\gamma} - \psi(\hat{\gamma}) = ln\bar{x} - \frac{1}{n}\sum_{i=1}^{n} lnx_i. \end{aligned} \tag{3.6}$$

The digamma function $\psi(\gamma)$ in Equation (3.6) can be calculated by Equation (3.7) (Thom, 1958)

$$\psi(\gamma) = ln(\gamma) - \frac{1}{2\gamma} - \frac{1}{12\gamma^2} + \frac{1}{120\gamma^4} - \frac{1}{252\gamma^6} + \frac{1}{240\gamma^8} - \frac{5}{660\gamma^8} + ... \tag{3.7}$$

or we can write in full asymptotic series as

$$\psi(\gamma) = ln(\gamma) - \frac{1}{2\gamma} + \sum_{n=1}^{m} \frac{B_{2n}}{2n}\gamma^{-2n} + R_m,$$

where

B_k are the Bernoulli numbers which is called a sequence of rational numbers denoted as $B_0 = 1, B_1 = \pm\frac{1}{2}, B_2 = -\frac{1}{6}, B_3 = 0, B_4 = -\frac{1}{30}$, etc. For all odd values of n other than 1, $B_n = 0$,

R_m is the remainder after m terms.

For $\gamma \geqslant 1$ we can write $|R_m|$ as (Thom, 1958)

$$|R_m| < -\frac{B_{m+3}}{(2m+2)\gamma^{2m+2}}.$$

In approximating the digamma function, Thom (1958) decided to use $m = 1$, $|R_m| < 0.0083$, and cancel the remainder function in approximating the digamma function. The approximation of digamma function becomes

$$\psi(\gamma) \approx ln\gamma - \frac{1}{2\gamma} - \frac{1}{12\gamma^2}. \tag{3.8}$$

Substituting digamma function as Equation (3.8) into Equation (3.6), we obtain,

$$\begin{aligned} ln\hat{\gamma} - \left[ln\hat{\gamma} - \frac{1}{2\hat{\gamma}} - \frac{1}{12\hat{\gamma}^2}\right] &= ln\bar{x} - \frac{1}{n}\sum_{i=1}^{n} lnx_i \\ \frac{1}{2\hat{\gamma}} + \frac{1}{12\hat{\gamma}^2} &= ln\bar{x} - \frac{1}{n}\sum_{i=1}^{n} lnx_i. \end{aligned} \tag{3.9}$$

Multiply Equation (3.9) by $12\hat{\gamma}^2$, we get

$$12\left(ln\bar{x} - \frac{1}{n}\sum_{i=1}^{n} lnx_i\right)\hat{\gamma}^2 - 6\hat{\gamma} - 1 = 0. \tag{3.10}$$

Simplifying Equation (3.10) by defining $A = ln\bar{x} - \frac{1}{n}\sum_{i=1}^{n} lnx_i$, then we have

$$12A\hat{\gamma}^2 - 6\hat{\gamma} - 1 = 0. \tag{3.11}$$

Solving quadratic Equation (3.11), we find two values for $\hat{\gamma}$, one is less than 0 and another is greater than 0. At the beginning, our condition for parameter gamma is $\gamma > 0$. So only when $\gamma > 0$ we can get the approximation of gamma as Equation (3.12)

$$\hat{\gamma} = \frac{1 + \sqrt{1 + \frac{4A}{3}}}{4A}, \tag{3.12}$$

where

$A = ln(\bar{x}) - \frac{1}{n}\sum_{i=1}^{n} lnx_i$,

n is number of precipitation observations,

x_i are the sample of rainfall data,

$\bar{x}$ is the mean of precipitation series.

The second derivative test maximum value for gamma distribution

There are two unknown parameters to compute the gamma distribution. They are shape parameter γ and scale parameter β. To test the gamma distribution achieve the maximum value at $\gamma = \hat{\gamma}$ and $\beta = \hat{\beta}$, the second derivative will be consider.

Let $\gamma = \hat{\gamma}$ and $\beta = \hat{\beta}$ be critical points of gamma distribution

$$\mathbb{D} = \left[f_{\beta\beta}(\hat{\gamma}, \hat{\beta})\right]\left[f_{\gamma\gamma}(\hat{\gamma}, \hat{\beta})\right] - \left[f_{\beta\gamma}(\hat{\gamma}, \hat{\beta})\right]^2,$$

where

$$f_{\beta\beta}(\hat{\gamma},\hat{\beta}) = \frac{n\hat{\gamma}}{\hat{\beta}^2} - \frac{2\sum_{i=1}^{n}}{\hat{\beta}^3},$$
$$f_{\gamma\gamma}(\hat{\gamma},\hat{\beta}) = \frac{-n}{\Gamma(\hat{\gamma})},$$
$$f_{\beta\gamma}(\hat{\gamma},\hat{\beta}) = \frac{-n}{\hat{\beta}},$$

n is the number of rainfall data.

If $\mathbb{D} > 0$ and $f_{\beta\beta}(\hat{\gamma},\hat{\beta}) < 0$, then $f(\hat{\gamma},\hat{\beta})$ has relative maximum at $\gamma = \hat{\gamma}$ and $\beta = \hat{\beta}$.

If $\mathbb{D} > 0$ and $f_{\beta\beta}(\hat{\gamma},\hat{\beta}) > 0$, then $f(\hat{\gamma},\hat{\beta})$ has relative minimum at $\gamma = \hat{\gamma}$ and $\beta = \hat{\beta}$.

Once the parameters γ and β have been estimated by using Equations (3.12) and (3.3), respectively. They are used to solve the cumulative probability density function of gamma distribution.

Cumulative probability density function of gamma distribution

The Cumulative Density Function (CDF) of gamma distribution is described by

$$\int_0^x f(x)dx = \frac{1}{\hat{\beta}^{\hat{\gamma}}\Gamma(\hat{\gamma})}\int_0^x x^{\hat{\gamma}-1}e^{\frac{-x}{\hat{\beta}}}dx. \tag{3.13}$$

Letting $t = \frac{x}{\hat{\beta}}$ and $\int_0^x f(x)dx = F(x)$, then Equation (3.13) becomes

$$\begin{aligned} F(x) &= \frac{1}{\hat{\beta}^{\hat{\gamma}}\Gamma(\hat{\gamma})} \int_0^x (t\hat{\beta})^{\hat{\gamma}-1} e^{-t}\hat{\beta}dt \\ &= \frac{1}{\hat{\beta}^{\hat{\gamma}}\Gamma(\hat{\gamma})} \int_0^x t^{\hat{\gamma}-1}\hat{\beta}^{\hat{\gamma}-1} e^{-t}\hat{\beta}dt \\ &= \frac{\hat{\beta}^{\hat{\gamma}}}{\hat{\beta}^{\hat{\gamma}}\Gamma(\hat{\gamma})} \int_0^x t^{\hat{\gamma}-1} e^{-t}dt \\ &= \frac{1}{\Gamma(\hat{\gamma})} \int_0^x t^{\hat{\gamma}-1} e^{-t}dt. \end{aligned} \tag{3.14}$$

As Equation (3.14), $\int_0^x t^{\hat{\gamma}-1}e^{-t}dt$ is incomplete gamma function $\tau(\gamma, t)$ which can be computed as $\tau(\gamma, t) = (\gamma - 1)!e^{-t}\sum_{k=0}^{\gamma-1}\frac{t^k}{k!}$. The gamma distribution is defined for $x > 0$, however, the actual precipitation can be 0. Thorn (1996) proposed the new cumulative probability of precipitation which accounts for zero. This function is given by

$$H(x) = u + (1 - u)F(x),$$

where

m is number of zero in the precipitation,

n is total number of precipitation series,

$u = \frac{m}{n}$ is probability of zero precipitation.

The cumulative probability of precipitation $H(x)$ is then transformed to a standard normal random variable z which has a mean of zero and variance of one. The random variable z is known as the SPI. An approximate value for SPI has

been given by Abramowitz and Stegun (1965).

$$SPI = \begin{cases} -k, & 0 < H(x) \leq 0.5 \\ \\ +k, & 0.5 < H(x) < 1, \end{cases}$$

where the values of k is introduced by Abramowitz and Stegun in 1965. The equation can be computed as

$$k = p - \frac{c_0 + c_1 p + c_2 p^2}{1 + d_1 p + d_2 p^2 + d_3 p^3},$$

constants c and d and variable p can be calculated as follows:

$$p = \begin{cases} \sqrt{ln\left(\frac{1}{H^2(x)}\right)}, & 0 < H(x) \leq 0.5 \\ \sqrt{ln\left(\frac{1}{(1-H(x))^2}\right)}, & 0.5 < H(x) < 1, \end{cases}$$

$$c_0 = 2.515517, c_1 = 0.802853, c_2 = 0.010328,$$

$$d_1 = 1.432788, d_2 = 0.189269, d_3 = 0.001308.$$

McKee *et al.* (1993) have defined drought intensities resulting from SPI value and also defined criteria for a drought event shown in Table 3.4.

3.4 Hopfield Networks

Hopfield network was introduced by Hopfield (1984). This network is bidirectional and fully connected with no-self connection (Hopfield, 1984). The structure is shown in Figure 3.5.

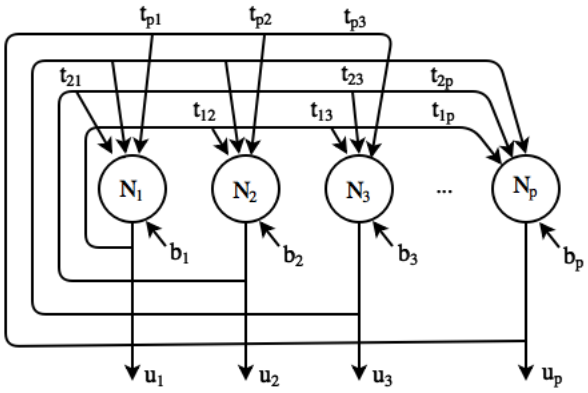

Figure 3.5: Hopfield network with p neurons and fully connected with no-self connection

Hopfield network in Figure 3.5 consists of p neurons, they are $N_1, N_2, N_3, ..., N_p$. This network is fully connected with symmetric weights t_{ij} which connected from neuron N_i to neuron N_j, $1 \leqslant i \leqslant p$ and $1 \leqslant j \leqslant p$. Furthermore, there is no-self connections ($t_{ii} = 0$). Each neuron in Hopfield network consists of two values, input value x_j and output value u_j (see Figure 3.6).

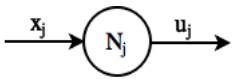

Figure 3.6: Single neuron N_j in Hopfield network with input value x_j and output value u_j

The input value x_j is calculated from sum of internal inputs which comes from other neurons and external input b_j, called bias value. The relationship between input value x_j and output value u_j is determined by a non-linear activation

function $g(x_j)$ of N_j. This activation function is a function used artificial neural network that introduce non-linear relationships between inputs and outputs of the artificial neural network. The input value x_j and output value u_j can be calculated from Equations (3.15) and (3.16), respectively,

$$x_j = \sum_{i=1, i \neq j}^{p} t_{ij} u_i + b_j, \tag{3.15}$$

$$u_j = g(x_j),$$

$$g(x_j) = \frac{1}{1 + e^{-x_j}} \tag{3.16}$$

where

x_j is the input value of N_j,

t_{ij} is the weight value from N_i to N_j,

b_j is the bias value of N_j,

u_j is the output value of N_j,

u_i is the output value of N_i,

$g(x_j)$ is the activation function of N_j.

There are two types of Hopfield networks. The first is binary Hopfield network which uses for binary data. The second is continuous Hopfield network which uses for continuous data. Any Hopfield network has one scalar value which associates with the state of the network, namely energy. The energy function of

binary Hopfield network is

$$E(u) = -\frac{1}{2}\sum_{\substack{i=1\\ i\neq j}}^{p}\sum_{j=1}^{p} u_i t_{ij} u_j - \sum_{j=1}^{p} b_j u_j, \tag{3.17}$$

where

u_i is the output value of N_i,

b_j is the bias value of N_j.

The energy function of continuous Hopfield network is

$$E(u) = -\frac{1}{2}\sum_{\substack{i=1\\ i\neq j}}^{p}\sum_{j=1}^{p} u_i t_{ij} u_j - \sum_{j=1}^{p} b_j u_j + \sum_{j=1}^{p}\frac{1}{R_j}\int_0^{u_j} g^{-1}(u)du, \tag{3.18}$$

where

R_j is the transmembrane resistance of N_j, $R_j > 0$,

$g^{-1}(u)$ is the inverse of activation function of N_j.

3.5 Restricted Boltzmann machines

Restricted Boltzmann machine (RBM) was introduced by Smolensky and Paul (1986). The network is made up of two layers. The first layer is visible layer which consists of visible neurons. The second layer is hidden layer which consists of hidden neurons. The structure of restricted Boltzmann machine is a complete bipartite graph in which all visible neurons are connected to all hidden neurons.

Figure 3.7 represents architecture of a restricted Boltzmann machine, the network consists of two layers. The first layer is visible layer which consists of m

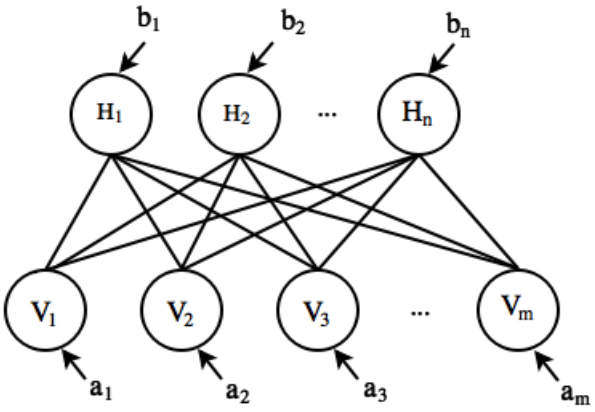

Figure 3.7: Restricted Boltzmann machine with visible neuron V_m and hidden neuron H_n

neurons, they are $V_1, V_2, V_3, ..., V_m$, each neuron has external input, namely bias value which is denoted by a_i, $1 \leqslant i \leqslant m$. The second layer is hidden layer which consists of n neurons, they are $H_1, H_2, H_3, ..., H_n$, each neuron has external input, namely bias value b_j, $1 \leqslant j \leqslant n$.

3.5.1 Single neuron of restricted Boltzmann machine

The structure of restricted Boltzmann machine consists of two layers, visible and hidden layers. A neuron in visible layer is called visible neuron and a neuron in hidden layer is called hidden neuron. Each visible neuron and hidden neuron associates with two values, which are input and output values.

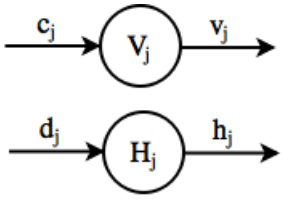

Figure 3.8: Single visible neuron V_j with input value c_j and output value v_j and single hidden neuron H_j with input value d_j and output value h_j

Figure 3.8 shows the input and output values of a visible neuron, V_j and the input and output values of a hidden neuron, H_j. Two values associated with visible neuron V_j are input value c_j and output value v_j. Two values associated with hidden neuron H_j, they are input value d_j and output value h_j.

The input value c_j for visible neuron V_j can be calculated as

$$c_j = \sum_{i=1}^{n} h_i w_{ij} + a_j, \tag{3.19}$$

where

h_i is the output value of H_i,

w_{ij} is the weight value from H_i to V_j,

a_j is the bias value of V_j.

Use the input value c_j as Equation (3.19) to calculate the output value v_j of visible neuron V_j as

$$v_j = \frac{1}{1 + e^{-c_j}}.$$

The input value d_j for the hidden neuron H_j can be calculated as

$$d_j = \sum_{i=1}^{m} v_i w_{ij} + b_j, \tag{3.20}$$

where

v_i is the output value of V_i,

w_{ij} is the weight value from V_i to H_j,

b_j is the bias value of H_j.

Use the input value d_j as Equation (3.20) to calculate the output value h_j for hidden neuron H_j as

$$h_j = \frac{1}{1 + e^{-d_j}}.$$

Restricted Boltzmann machine is an energy-based model. Therefore, the energy function of restricted Boltzmann machine will be explained in Section 3.5.2.

3.5.2 Energy function of restricted Boltzmann machine

Restricted Boltzmann machine is an energy-based model with associated scalar value, namely energy. Since the restricted Boltzmann machine involves with the variables of v and h, it is appropriate to use join configuration of those variables. Let $v \in \mathbb{R}^m$ and $h \in \mathbb{R}^n$ where $v = (v_1, v_2, v_3, ..., v_m)$ and $h = (h_1, h_2, h_3, ..., h_n)$. Therefore, the energy function of restricted Boltzmann machine can be computed as

$$E(v, h) = -\sum_{i=1}^{m}\sum_{j=1}^{n} v_i w_{ij} h_j - \sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j, \tag{3.21}$$

where

v_i is the output value of V_i,

w_{ij} is the weight values from V_i to H_j,

h_j is the output value of H_j,

a_i is the bias value of V_i,

b_j is the bias value of H_j.

The details of mathematical derivation of the energy function of restricted Boltzmann machine may be seen in appendix A.

3.5.3 Minimizing contrastive divergence for RBM

Restricted Boltzmann machine can be trained using the minimizing contrastive divergence (MCD) algorithm (Hinton, 2002). To train restricted Boltzmann machine, there are three unknown parameters, weight w_{ij}, bias a_i, and bias b_j. Therefore, the training rule for any parameter θ of restricted Boltzmann machine by using the minimizing contrastive divergence is

$$\Delta\theta = \left\langle -\frac{\partial}{\partial\theta}E(v,h)\right\rangle_0 - \left\langle -\frac{\partial}{\partial\theta}E(v,h)\right\rangle_\infty ,$$

where

$\langle . \rangle_0$ is the expectation value over the training data with visible state clamped,

$\langle . \rangle_\infty$ is the expectation value over the training data in free-running equilibrium.

The intuitive motivation to use minimizing contrastive divergence is using one-step Gibbs sampling to estimate the expectation value of $-\frac{\partial}{\partial\theta_i}E(v,h)$ over the training data in free-running equilibrium ($\left\langle -\frac{\partial}{\partial\theta_i}E(v,h)\right\rangle_\infty$). Thus, the training rule for any parameter θ becomes

$$\Delta\hat{\theta} = \left\langle -\frac{\partial}{\partial\theta}E(v,h)\right\rangle_0 - \left\langle -\frac{\partial}{\partial\theta}E(v,h)\right\rangle_1 ,$$

where

$\langle . \rangle_1$ is the expectation value over the training data after one-step Gibbs sampling.

As mentioned before, there are three unknown parameters which will be updated while training restricted Boltzmann machine. They are weight w_{ij}, bias a_i, and bias b_j. The training rule of weight parameter w_{ij} is

$$\Delta \hat{w}_{ij} = \left\langle -\frac{\partial}{\partial w_{ij}} E(v,h) \right\rangle_0 - \left\langle -\frac{\partial}{\partial w_{ij}} E(v,h) \right\rangle_1 . \tag{3.22}$$

To get the training rule for weight parameter, we have to find the negative derivative of energy function with respect to w_{ij}. We obtain

$$-\frac{\partial}{\partial w_{ij}} E(v,h) = \frac{\partial}{\partial w_{ij}} \sum_{k=1}^{m} \sum_{l=1}^{n} v_k w_{kl} h_l + \frac{\partial}{\partial w_{ij}} \sum_{k=1}^{m} a_k v_k + \frac{\partial}{\partial w_{ij}} \sum_{l=1}^{n} b_l h_l.$$

The derivative of $\frac{\partial}{\partial w_{ij}} \sum_{k=1}^{m} \sum_{l=1}^{n} v_k w_{kl} h_l$ is zero for all terms in the summation except $k = i$ and $l = j$, hence,

$$\begin{aligned} -\frac{\partial}{\partial w_{ij}} E(v,h) &= \frac{\partial}{\partial w_{ij}} v_i w_{ij} h_j \\ &= v_i h_j. \end{aligned} \tag{3.23}$$

Let

$v_i^{(0)}$ be the output value of V_i at the initial state,

$v_i^{(1)}$ be the output value of V_i at the one-step Gibb sampling state,

$h_j^{(0)}$ be the output value of H_j at the initial state,

$h_j^{(1)}$ be the output value of H_j at the one-step Gibb sampling state.

Substituting Equation (3.23) in Equation (3.22), we obtain the training rule of weight parameter w_{ij} as

$$\Delta\hat{w}_{ij} = \left\langle v_i^{(0)} h_j^{(0)} \right\rangle - \left\langle v_i^{(1)} h_j^{(1)} \right\rangle .$$

For the bias a_i, the training rule is

$$\Delta\hat{a}_i = \left\langle -\frac{\partial}{\partial a_i} E(v,h) \right\rangle_0 - \left\langle -\frac{\partial}{\partial a_i} E(v,h) \right\rangle_1 . \tag{3.24}$$

From Equation (3.24), the negative derivative of energy function with respect to the parameter a_i is

$$-\frac{\partial}{\partial a_i} E(v,h) = \frac{\partial}{\partial a_i} \sum_{k=1}^{m} \sum_{l=1}^{n} v_k w_{kl} h_l + \frac{\partial}{\partial a_i} \sum_{k=1}^{m} a_k v_k + \frac{\partial}{\partial a_i} \sum_{l=1}^{n} b_l h_l$$

The derivative of $\frac{\partial}{\partial a_i} \sum_{k=1}^{m} a_k v_k$ is zero for all terms in the summation except $k = i$, hence,

$$\begin{aligned} -\frac{\partial}{\partial a_i} E(v,h) &= \frac{\partial}{\partial a_i} a_i v_i \\ &= v_i. \end{aligned} \tag{3.25}$$

Substituting Equation (3.25) in Equation (3.24), we obtain the training rule for bias parameter a_i as

$$\Delta\hat{a}_i = \left\langle v_i^{(0)} \right\rangle - \left\langle v_i^{(1)} \right\rangle .$$

The training rule for bias b_j is

$$\Delta\hat{b}_j = \left\langle -\frac{\partial}{\partial b_j} E(v,h) \right\rangle_0 - \left\langle -\frac{\partial}{\partial b_j} E(v,h) \right\rangle_1 . \tag{3.26}$$

The negative derivative of energy function with respect to the parameter b_j is

$$-\frac{\partial}{\partial b_j} E(v,h) = \frac{\partial}{\partial b_j} \sum_{k=1}^{m} \sum_{l=1}^{n} v_k w_{kl} h_l + \frac{\partial}{\partial b_j} \sum_{k=1}^{m} a_k v_k + \frac{\partial}{\partial b_j} \sum_{l=1}^{n} b_l h_l.$$

The derivative of $\frac{\partial}{\partial b_j} \sum_{l=1}^{n} b_l h_l$ is zero for all terms in the summation except $l = j$, hence,

$$\begin{aligned} -\frac{\partial}{\partial b_j} E(v,h) &= \frac{\partial}{\partial b_j} b_j h_j \\ &= h_j. \end{aligned} \tag{3.27}$$

Substituting Equation (3.27) in Equation (3.26), we obtain the training rule for bias parameter b_j as

$$\Delta\hat{b}_j = \left\langle h_i^{(0)} \right\rangle - \left\langle h_i^{(1)} \right\rangle .$$

In practice, when using the MCD algorithm to train these three parameters, weight w_{ij}, bias a_i, and bias b_j are incrementally updated to achieve the final value. The training of each parameter will be updated by assigning a learning rate to each of training rule. The learning rate is a real number between 0 to 1, this value can be defined by users.

The training rule of each parameter combining with the learning rate establish the update rule for each parameter in the minimizing contrastive divergence algorithm as follows:

$$\Delta\hat{w}_{ij} = \eta_w \left\langle v_i^{(0)} h_j^{(0)} \right\rangle - \left\langle v_i^{(1)} h_j^{(1)} \right\rangle,$$
$$\Delta\hat{a}_i = \eta_a \left\langle v_i^{(0)} \right\rangle - \left\langle v_i^{(1)} \right\rangle,$$
$$\Delta\hat{b}_j = \eta_b \left\langle h_j^{(0)} \right\rangle - \left\langle h_j^{(1)} \right\rangle.$$

where

η_w is the learning rate for all weight parameters,

η_a is the learning rate for all bias value in V_i,

η_b is the learning rate for all bias value in H_j.

To compute the expectation value of each parameter in restricted Boltzmann machine, we can compute by using Gibbs sampling (see appendix C).

3.6 Continuous restricted Boltzmann machines

A continuous restricted Boltzmann machine (CRBM) was introduced by Chen and Murray (2003). It is a form of restricted Boltzmann machine with no bias value, which can model continuous data. The structure of continuous restricted Boltzmann machine is shown in Figure 3.9.

Figure 3.9 is the architecture of continuous restricted Boltzmann machine which consists of two layers. The first layer is visible layer consists of m neurons, they are $V_1, V_2, V_3, ..., V_m$. The second layer is hidden layer. They are

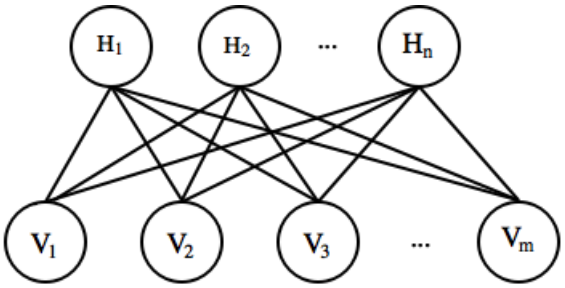

Figure 3.9: Continuous restricted Boltzmann machine with visible neuron V_j and hidden neuron H_j

$H_1, H_2, H_3, ..., H_n$, it is n neurons.

3.6.1 Single neuron of continuous restricted Boltzmann machine

The structure of continuous restricted Boltzmann machine consists of two layers, they are visible layer and hidden layer. A neuron in visible layer is called visible neuron and a neuron in hidden layer is called hidden neuron. Each visible neuron and hidden neuron associates with two values which are input value and output value.

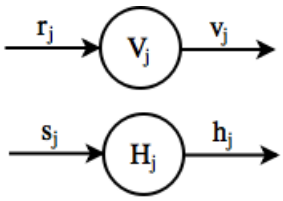

Figure 3.10: Single visible neuron V_j with input value r_j and output value v_j and single hidden neuron H_j with input value s_j and output value h_j

Figure 3.10 shows the input and output values of a visible neuron, V_j and the input and output values of hidden neuron, H_j. The visible neuron V_j associates with two values, input value r_j and output value v_j. The hidden neuron H_j associates with two values, they are input value s_j and output value h_j.

To model continuous data, Chen *et al.* (2003) modified a binary unit into

continuous stochastic unit by adding zero-mean Gaussian with unit variance into the input of sigmoid unit. Therefore, the input value r_j of visible neuron V_j can be computed as

$$r_j = \sum_{i=1}^{n} w_{ij} h_i + \sigma N_j(0,1), \tag{3.28}$$

where

w_{ij} is the weight value from H_i to V_j,

h_i is the output value of H_i,

σ is the constant,

$N_j(0,1)$ is the Gaussian random variable with mean zero and unit variance associated with V_j.

The constant σ can be set by users which the number is real number. Commonly, this number is located between $0-1$.

The input value r_j as Equation (3.28) will be used to compute the output value v_j of visible neuron V_j as

$$v_j = \varphi(r_j),$$

with

$$\varphi(r_j) = \theta_L + (\theta_H - \theta_L)\frac{1}{1+e^{-\kappa_j r_j}}, \tag{3.29}$$

where

v_j is the output value of V_j,

r_j is the input value of V_j,

$\varphi(r_j)$ is the activation function of V_j,

θ_H, θ_L are constant upper and lower asymptotes,

κ_j is noise-control parameter of V_j, $\kappa_j > 0$.

There are two values for hidden neuron H_j, which are input value s_j and output value h_j. The input value s_j can be computed as

$$s_j = \sum_{i=1}^{m} w_{ij} v_i + \sigma N_j(0,1), \tag{3.30}$$

where

w_{ij} is the weight value from V_i to H_j,

v_i is the output value of V_i,

σ is the constant,

$N_j(0,1)$ is the Gaussian random variable with mean zero and unit variance associated with H_j.

Use Equation (3.30) to compute the output value h_j of hidden neuron H_j. Follow Equation (3.31)

$$h_j = \psi(s_j),$$

with

$$\psi(s_j) = \theta_L + (\theta_H - \theta_L)\frac{1}{1 + e^{-\lambda_j s_j}}, \tag{3.31}$$

where

h_j is the output value of H_j,

s_j is the input value of H_j,

$\psi(s_j)$ is the activation function of H_j,

θ_H, θ_L are constant upper and lower asymptotes,

λ_j is noise-control parameter of H_j, $\lambda_j > 0$.

3.6.2 Energy function of continuous restricted Boltzmann machine

The energy function of continuous restricted Boltzmann machine can be computed as Equation (3.32)

$$\begin{aligned} E(v,h) = &-\sum_{i=1}^{m}\sum_{j=1}^{n} v_i w_{ij} h_j + \sum_{i=1}^{m} \frac{1}{\kappa_i} \int_0^{v_i} \varphi^{-1}(v_i) dv \\ &+ \sum_{j=1}^{n} \frac{1}{\lambda_j} \int_0^{h_j} \psi^{-1}(h_j) dh, \end{aligned} \tag{3.32}$$

where

v_i is the output value of V_i,

w_{ij} is the weight value from V_i to H_j,

h_j is the output value of H_j,

κ_i is the value of noise control parameter of V_i, $\kappa_i > 0$,

$\varphi^{-1}(v)$ is the inverse of the activation function of V_i,

λ_j is the value of noise control parameter of H_j, $\lambda_j > 0$,

$\psi^{-1}(h)$ is the inverse of the activation function of H_j.

The mathematical derivation to get the energy function of continuous restricted Boltzmann machine is shown in appendix B.

3.6.3 Minimizing contrastive divergence for CRBM

To train continuous restricted Boltzmann machine, there are three unknown parameters, weight parameter w_{ij}, noise control parameter κ_i of visible neuron V_i, and noise control parameter λ_j of hidden neuron H_j.

A continuous restricted Boltzmann machine is actual a form of symmetrical restricted diffusion network. Therefore, the training rule of each parameter in diffusion network is appropriated to train continuous restricted Boltzmann machine. The training rule for any parameter θ in diffusion network is shown in Equation (3.33)

$$\Delta\theta = \langle S_\theta \rangle_0 - \langle S_\theta \rangle_\infty , \tag{3.33}$$

where

$\langle . \rangle_0$ is the expectation value over the training data with visible state clamped,

$\langle . \rangle_\infty$ is the expectation value over the training data in free-running equilibrium,

S_θ is system covariances.

The intuitive motivation to use minimizing contrastive divergence is using one step Gibbs sampling to estimate the expectation value of S_θ over the training data in free-running equilibrium, $\langle S_\theta \rangle_\infty$. Therefore, the training rule for any parameter θ of diffusion network using one step Gibbs sampling is

$$\Delta\hat{\theta} = \langle S_\theta \rangle_0 - \langle S_\theta \rangle_1 ,$$

where $\langle S_\theta \rangle_1$ represents the expectation value of S_θ over the training data after one-step Gibb sampling.

For a diffusion network, the system covariances S_θ is negative derivative of the energy function with respect to θ. Therefore, the training rule for any parameter θ of continuous restricted Boltzmann machine becomes

$$\Delta\hat{\theta} = \left\langle -\frac{\partial}{\partial\theta}E(v,h) \right\rangle_0 - \left\langle -\frac{\partial}{\partial\theta}E(v,h) \right\rangle_1 . \tag{3.34}$$

As mentioned before, training continuous restricted Boltzmann machine involves estimating three unknown parameters, they are w_{ij}, κ_i, and λ_j. From Equation (3.34), we can derive the training rule for weight parameter w_{ij} as

$$\Delta\hat{w}_{ij} = \left\langle -\frac{\partial}{\partial w_{ij}}E(v,h) \right\rangle_0 - \left\langle -\frac{\partial}{\partial w_{ij}}E(v,h) \right\rangle_1 . \tag{3.35}$$

To get the training rule of weight parameter, we have to differentiate the negative of energy function with respect to w_{ij},

$$\begin{aligned} -\frac{\partial}{\partial w_{ij}}E(v,h) = & \frac{\partial}{\partial w_{ij}}\sum_{k=1}^{m}\sum_{l=1}^{n} v_k w_{kl} h_l \\ & - \frac{\partial}{\partial w_{ij}}\sum_{k=1}^{m}\frac{1}{\kappa_k}\int_0^{v_k}\varphi^{-1}(v_k)dv \\ & - \frac{\partial}{\partial w_{ij}}\sum_{l=1}^{n}\frac{1}{\lambda_l}\int_0^{h_l}\psi^{-1}(h_l)dh. \end{aligned}$$

The derivatives $\frac{\partial}{\partial w_{ij}} \sum_{k=1}^{m} \sum_{l=1}^{n} v_k w_{kl} h_l$ are zero for all terms in the summation except $k = i$ and $l = j$, hence

$$-\frac{\partial}{\partial w_{ij}} E(v, h) = \frac{\partial}{\partial w_{ij}} v_i w_{ij} h_j = v_i h_j. \tag{3.36}$$

Substituting Equation (3.36) in Equation (3.35). Then, we obtain the training rule for weight parameter w_{ij} as

$$\Delta \hat{w}_{ij} = \langle v_i h_j \rangle_0 - \langle v_i h_j \rangle_1 .$$

Let

$v_i^{(0)}$ be the output value of V_i at the initial state,

$v_i^{(1)}$ be the output value of V_i at the one-step Gibb sampling state,

$h_j^{(0)}$ be the output value of H_j at the initial state,

$h_j^{(1)}$ be the output value of H_j at the one-step Gibb sampling state.

Use new notations as defined above, the training rule for weight parameter w_{ij} becomes

$$\Delta \hat{w}_{ij} = \left\langle v_i^{(0)} h_j^{(0)} \right\rangle - \left\langle v_i^{(1)} h_j^{(1)} \right\rangle .$$

Using the same process of the training rule for wight parameter to get the training rule of κ_i for visible neuron V_i.

$$\Delta \hat{\kappa}_i = \left\langle -\frac{\partial}{\partial \kappa_i} E(v, h) \right\rangle_0 - \left\langle -\frac{\partial}{\partial \kappa_i} E(v, h) \right\rangle_1 . \tag{3.37}$$

The negative derivative of energy function with respect to κ_i is

$$\begin{aligned} -\frac{\partial}{\partial \kappa_i} E(v,h) = & \frac{\partial}{\partial \kappa_i} \sum_{k=1}^{m} \sum_{l=1}^{n} v_k w_{kl} h_l \\ & - \frac{\partial}{\partial \kappa_i} \sum_{k=1}^{m} \frac{1}{\kappa_k} \int_0^{v_k} \varphi^{-1}(v_k) dv \\ & - \frac{\partial}{\partial \kappa_i} \sum_{l=1}^{n} \frac{1}{\lambda_l} \int_0^{h_l} \psi^{-1}(h_l) dh. \end{aligned}$$

The derivative $\frac{\partial}{\partial \kappa_i} \sum_{k=1}^{m} \frac{1}{\kappa_k} \int_0^{v_k} \varphi^{-1}(v_k) dv$ is zero for all terms in the summation except when $k = i$. Therefore, the derivative becomes

$$\begin{aligned} -\frac{\partial}{\partial \kappa_i} E(v,h) &= \left(- \int_0^{v_i} \varphi^{-1}(v_i) dv \right) \left(\frac{\partial}{\partial \kappa_i} \frac{1}{\kappa_i} \right) \\ &= \frac{1}{\kappa_i^2} \int_0^{v_i} \varphi^{-1}(v_i) dv. \end{aligned} \tag{3.38}$$

Substituting Equation (3.38) in Equation (3.37), we obtain the training rule of noise control parameter κ_i as

$$\Delta \hat{\kappa}_i = \left\langle \frac{1}{\kappa_i^2} \int_0^{v_i} \varphi^{-1}(v_i) dv \right\rangle_0 - \left\langle \frac{1}{\kappa_i^2} \int_0^{v_i} \varphi^{-1}(v_i) dv \right\rangle_1 . \tag{3.39}$$

By the property of expectation value, if X is random variable and c is constant, we have expectation value $\mathbb{E}(cX) = c\mathbb{E}(X)$. When consider the expectation value over the training data as Equation (3.39), v_i is a random variable and $\frac{1}{{\kappa_i}^2}$ is

a constant. Therefore, we obtain

$$\begin{aligned}\Delta\hat{\kappa}_i &= \frac{1}{\kappa_i^2}\left\langle\int_0^{v_i}\varphi^{-1}(v_i)dv\right\rangle_0 - \frac{1}{\kappa_i^2}\left\langle\int_0^{v_i}\varphi^{-1}(v_i)dv\right\rangle_1 \\ &= \frac{1}{\kappa_i^2}\left[\left\langle\int_0^{v_i}\varphi^{-1}(v_i)dv\right\rangle_0 - \left\langle\int_0^{v_i}\varphi^{-1}(v_i)dv\right\rangle_1\right] \\ &= \frac{1}{\kappa_i^2}\left[\left\langle\int_0^{v_i^{(0)}}\varphi^{-1}(v_i)dv\right\rangle - \left\langle\int_0^{v_i^{(1)}}\varphi^{-1}(v_i)dv\right\rangle\right]\end{aligned} \tag{3.40}$$

Using the property of expectation value, if X and Y are random variables, we have expectation value $\mathbb{E}(X+Y) = \mathbb{E}(X) + \mathbb{E}(Y)$. As Equation (3.40), we obtain

$$\begin{aligned}\Delta\hat{\kappa}_i &= \frac{1}{\kappa_i^2}\left[\left\langle\int_0^{v_i^{(0)}}\varphi^{-1}(v_i)dv - \int_0^{v_i^{(1)}}\varphi^{-1}(v_i)dv\right\rangle\right] \\ &= \frac{1}{\kappa_i^2}\left[\left\langle\int_{v_i^{(1)}}^{v_i^{(0)}}\varphi^{-1}(v_i)dv\right\rangle\right].\end{aligned} \tag{3.41}$$

The integral term in Equation (3.41) has been approximated by Chen *et al.* (2003) as shown in Equation (3.42)

$$\int_{v_i^{(1)}}^{v_i^{(0)}}\varphi^{-1}(v_i)dv \approx (v_i^{(0)} + v_i^{(1)})(v_i^{(0)} - v_i^{(1)}). \tag{3.42}$$

Substituting Equation (3.42) in Equation (3.41), we obtain the training rule for noise control parameter κ_i as

$$\Delta\hat{\kappa}_i = \frac{1}{\kappa_i^2}\left(\left\langle\left(v_i^{(0)}\right)^2\right\rangle - \left\langle\left(v_i^{(1)}\right)^2\right\rangle\right).$$

Similarly for the noise control parameter λ_j of hidden neuron H_j, we obtain the training rule as

$$\Delta\hat{\lambda}_j = \frac{1}{\lambda_j^2}\left(\left\langle\left(h_j^{(0)}\right)^2\right\rangle - \left\langle\left(h_j^{(1)}\right)^2\right\rangle\right).$$

The MCD algorithm uses the training rules in combination with learning rates to incrementally update the parameter values to achieve the final values. Therefore, we obtain the update rule for weight w_{ij}, noise control parameter κ_i for visible neuron V_i, and noise control parameter λ_j for hidden neuron H_j, which are shown in Equations (3.43), (3.44), and (3.45), respectively,

$$\Delta\hat{w}_{ij} = \eta_w\left(\left\langle v_i^{(0)}h_j^{(0)}\right\rangle - \left\langle v_i^{(1)}h_j^{(1)}\right\rangle\right), \tag{3.43}$$

$$\Delta\hat{\kappa}_i = \frac{\eta_\kappa}{\kappa_i^2}\left(\left\langle {v_i^{(0)}}^2\right\rangle - \left\langle {v_i^{(1)}}^2\right\rangle\right), \tag{3.44}$$

$$\Delta\hat{\lambda}_j = \frac{\eta_\lambda}{\lambda_j^2}\left(\left\langle {h_j^{(0)}}^2\right\rangle - \left\langle {h_j^{(1)}}^2\right\rangle\right), \tag{3.45}$$

where

η_w is the learning rate for all weight parameters,

η_κ is the learning rate for all noise control parameters of visible neurons V_j,

η_λ is the learning rate for all noise control parameters of hidden neurons H_j.

3.6.4 The MCD algorithm for continuous restricted Boltzmann machine

The algorithm uses to update the parameters of continuous restricted Boltzmann machine using the minimizing contrastive divergence is shown in Algorithm 1.

The Algorithm 1 is illustrated the process to compute the update rule of w_{ij}, κ_i, and λ_j in continuous restricted Boltzmann machine. This algorithm works under the process of one-step Gibbs sampling. The input data of the first one-step Gibbs sampling are $x_1, x_2, x_3, ..., x_m$, where m is the number of input nodes. Using the algorithm 1 to learn this data set, the parameters of w_{ij}, κ_i, and λ_j will be update. Next, the input data of the second one-step Gibbs sampling are $x_2, x_3, ..., x_{m+1}$. Using the Algorithm 1 to learn this data set, those parameters will be updated again. This process will repeat until the algorithm learns all data.

Algorithm 1 MCD algorithm for continuous restricted Boltzmann machine

procedure MCD($x_1, x_2, x_3, ..., x_m$) ▷ m is the number of input nodes
  for `i=1 to m` **do**
    $v_i^{(0)} = x_i$
  end for
  for `i=1 to m` **do**
    for `j=1 to n` **do**
      randomize w_{ij} ▷ Initialize w_{ij} with random values
    end for
  end for
  for `t=0 to K` **do** ▷ One-step Gibbs sampling
    for `j=1 to n` **do** ▷ Positive phase
      $s_j^{(t)} = \sum_{i=1}^{m} w_{ij} v_i + \sigma N_j(0,1)$
      $h_j^{(t)} = \theta_L + (\theta_H - \theta_L) \frac{1}{1+e^{-\lambda_j s_j}}$
    end for
    if t $\neq$ K **then**
      for `j=1 to n` **do** ▷ Negative phase
        $r_j^{(t+1)} = \sum_{i=1}^{n} w_{ij} h_i + \sigma N_j(0,1)$
        $v_j^{(t+1)} = \theta_L + (\theta_H - \theta_L) \frac{1}{1+e^{-\kappa_j r_j}}$
      end for
    end if
  end for
  for `i=1 to m` **do**
    for `j=1 to n` **do**
      $wpos_{ij} = v_i^{(0)} h_j^{(0)}$
      $wneg_{ij} = v_i^{(K)} h_j^{(K)}$
      $\Delta\hat{w}_{ij} = \eta_w \left(wpos_{ij} - wneg_{ij} \right)$
      $w_{ij_{new}} = w_{ij_{old}} - \Delta\hat{w}_{ij}$ ▷ Update w_{ij}
    end for
  end for
  for `i=1 to m` **do**
    $kappapos_i = \left(v_i^{(o)}\right)^2$
    $kappaneg_i = \left(v_i^{(K)}\right)^2$
    $\Delta\hat{\kappa}_i = \frac{\eta_\kappa}{\kappa_i^2} \left(kappapos_i - kappaneg_i \right)$
    $\kappa_{i_{new}} = \kappa_{i_{old}} - \Delta\hat{\kappa}_i$ ▷ Update κ_i
  end for
  for `j=1 to n` **do**
    $lambdapos_j = \left(h_j^{(o)}\right)^2$
    $lambdaneg_j = \left(h_j^{(K)}\right)^2$
    $\Delta\hat{\lambda}_j = \frac{\eta_\lambda}{\lambda_j^2} \left(lambdapos_j - lambdaneg_j \right)$
    $\lambda_{j_{new}} = \lambda_{j_{old}} - \Delta\hat{\lambda}_j$ ▷ Update λ_j
  end for
end procedure

3.7 Feedforward neural network

Feedforward neuron network (FFNN) is an artificial neural network (ANN) in which the structure consists of three of types layers, they are input layer, hidden layer, and output layer.

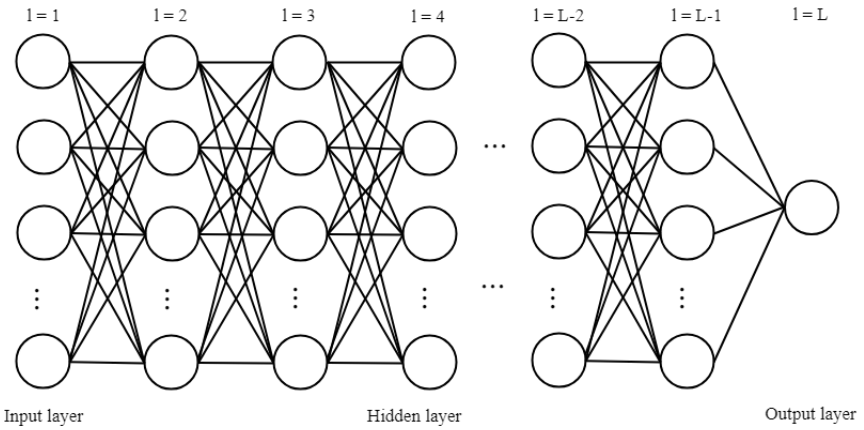

Figure 3.11: Feedforward neuron network with L layers

Figure 3.11 shows the structure of feedforward neuron network for regression problem. The first layer is input layer, denoted $l = 1$, which consists of input nodes. The last layer is output layer, denoted $l = L$, which consists of one output neuron. The hidden layers are all layers between input and output layers, denoted $2 \leqslant l \leqslant L - 1$. There may be many hidden layers and each hidden layer consists of hidden neurons. The number of hidden neurons in each hidden layer can be different number. There is an weight value attached to any edge in this graph, namely w_{ij}^{l} and w_{i}^{L}. When w_{ij}^{l} denoted the weight value of the edge from neuron i at layer $l - 1$ to neuron j at layer l $(2 \leqslant l \leqslant L - 1)$ and w_{i}^{L} denoted the weight value of the edge from neuron i at layer $L - 1$ to output neuron. Also, let M^{l} denotes the number of input nodes/neurons in layer l, $1 \leqslant l \leqslant L$.

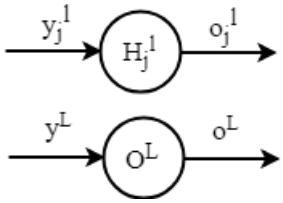

Figure 3.12: Single hidden neuron H_j^l and single output neuron O^L in FFNN

Figure 3.12 shows the input and output values of hidden neuron j at layer l, H_j^l, and the input and output values of output neuron at layer L, O^L.

The hidden neuron H_j^l associates with two values, input value y_j^l and output value o_j^l in which

y_j^l is the input value of hidden neuron j at layer l,

o_j^l is the output value of hidden neuron j at layer l.

The output neuron O^L associates with two values, input value y^L and output value o^L in which

y^L is the input value of output neuron at layer L,

o^L is the output value of output neuron at layer L.

3.7.1 Back-propagation neural network algorithm

Back-propagation neural network algorithm is a common method to train neuron networks. In this research, this algorithm will be used to train feedforward neuron networks. Back-propagation neuron network algorithm looks for the weights values which minimize error using gradient descent method. It involves minimizing the error as shown in Equation (3.46)

$$E^L(t, o) = \frac{1}{2}(t - o^L)^2, \tag{3.46}$$

where

E^L is the error function in layer L,

t is the target value,

o^L is the output value of output neuron in layer L.

To learn the weight values by using the gradient descent method, the derivative of error function will be considered. Therefore, the training rule of weight parameter (w_{ij}^l) from neuron i at layer $l-1$ to neuron j at layer l, Δw_{ij}^l, is

$$\Delta w_{ij}^l = -\frac{\partial E^L}{\partial w_{ij}^l}, \tag{3.47}$$

In order to get the appropriated weight values using back-propagation neuron network algorithm, two steps are required. The first step is forward propagation and the second step is backward propagation.

Step 1: forward propagation

Feedforward neuron network consists of three of type layers, input layer, hidden layer, and output layer. Forward propagation is the processes of getting the value of each input node in input layer ($l = 1$) from data. Furthermore, this process is also computing the input and output values of each hidden neuron in hidden layer $2 \leqslant l \leqslant L-1$ in order and output neuron in output layer at layer L. The process to obtain these values is described as follows:

The value of input node in input layer at layer $l = 1$

At the beginning we will work on the input node in input layer at layer $l = 1$. The input value of each input node in input layer can be taken directly from the data and the output value of the input node is simply the input value. Let x_i denote the input or output values of input node i at layer $l = 1$.

After getting the output value of each input node in input layer $l = 1$, we will compute the input and output values for each hidden neuron in hidden layer $l = 2$ to $l = L - 1$ in order.

Computing the input and output values of hidden neuron H_j^l when $2 \leqslant l \leqslant L - 1$

At the beginning of this process, we will compute the input value y_j^2 and output value o_j^2 of hidden neuron H_j^2. The equations to compute are shown as follows.

$$y_j^2 = \sum_{i=1}^{M^1} {w_{ij}}^2 x_i^{l-1} + b_j^2, \tag{3.48}$$

$$o_j^2 = \frac{1}{1 + e^{(-y_j^2)}}. \tag{3.49}$$

where

y_j^2 is the input value of hidden neuron H_j^2,

w_{ij}^2 is the weight value from input node i in input layer to hidden neuron H_j^2,

x_i is the output value of input node i in input layer,

M^1 is the number of input node in input layer,

b_j^2 is the bias value of H_j^2,

o_j^2 is the output value of H_j^2.

Next, we will compute the input value y_j^l and output value o_j^l for each hidden neuron H_j^l in layer $l = 3$. Then, we will move to compute input value y_j^l and output value o_j^l for each hidden neuron H_j^l in layer $l = 4$. We will compute the input value y_j^l and output value o_j^l for each hidden neurons H_j^l until layer $l = L - 1$. The equations to compute these values are shown as follows:

$$y_j^l = \sum_{i=1}^{M^{l-1}} {w_{ij}}^l o_i^{l-1} + b_j^l, \tag{3.50}$$

$$o_j^l = \frac{1}{1 + e^{(-y_j^l)}} \tag{3.51}$$

where

y_j^l is the input value of hidden neuron H_j^l ($1 \leqslant j \leqslant M^l$, $3 \leqslant l \leqslant L - 1$),

w_{ij}^l is the weight value from hidden neuron H_i^{l-1} to hidden neuron H_j^l ($1 \leqslant i \leqslant M^{l-1}$, $1 \leqslant j \leqslant M^l$, $3 \leqslant l \leqslant L - 1$),

o_i^{l-1} is the output value of hidden neuron H_i^{l-1} ($1 \leqslant i \leqslant M^{l-1}$, $3 \leqslant l \leqslant L - 1$) ,

M^{l-1} is the number of hidden neuron in layer $l - 1$ ($3 \leqslant l \leqslant L - 1$),

b_j^l is the bias value of hidden neuron H_j^l ($1 \leqslant j \leqslant M^l$, $3 \leqslant l \leqslant L - 1$),

o_j^l is the output value of hidden neuron H_j^l ($1 \leqslant j \leqslant M^l$, $3 \leqslant l \leqslant L - 1$).

After computing the input and output values of all hidden neurons in each hidden layer, we will compute the input and output values of the output neuron in the next step.

Computing the input and output values of the output neuron O^L at layer L

The last process of Step 1 is computing the input value y^L and output value o^L of output neuron O^L at layer L which can be computed as

$$y^L = \sum_{i=1}^{M^{L-1}} w_i^L o_i^{L-1} + b^L, \tag{3.52}$$

$$o^L = y^L \tag{3.53}$$

where

y^L is the input value of the output neuron at layer L,

w_i^L is the weight value from hidden neuron H_i^{L-1} to output neuron O ($1 \leqslant i \leqslant M^{L-1}$),

o_i^{L-1} is the output value of hidden neuron H_i^{L-1} ($1 \leqslant i \leqslant M^{L-1}$),

M^{L-1} is the number of hidden neuron in layer $L-1$,

b^L is the bias value of the output neuron in layer L,

o^L is the output value of the output neuron at layer L.

Step 2: backward propagation

Backward propagation is the process of adjusting the weight values by minimizing the error function. There are two cases for updating the weight values.

The first case is updating the weight values for output neuron O^L (w_i^L, $1 \leqslant i \leqslant M^{L-1}$) and the second case is updating the weight values for hidden neuron H_j^l (w_{ij}^l, $2 \leqslant l \leqslant L-1$, $1 \leqslant i \leqslant M^{l-1}$, $1 \leqslant j \leqslant M^l$).

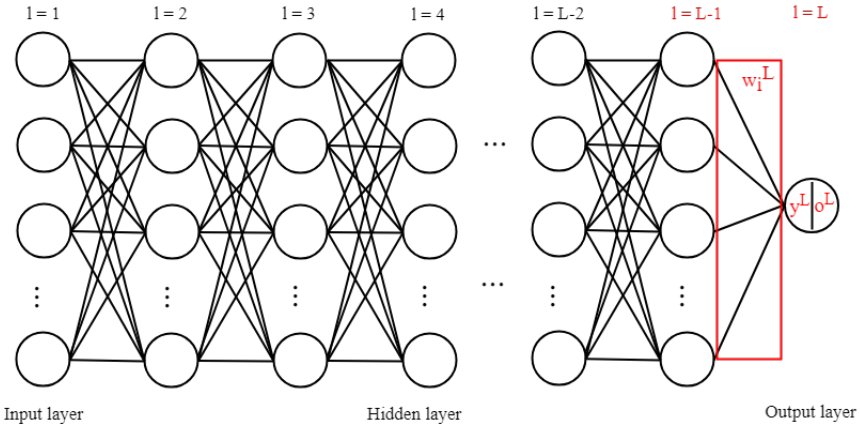

Figure 3.13: Updating the weight values for output neuron at layer L

Case 1: Updating the weight values for output neuron at layer L

To update the weight value from hidden neuron i at layer $L-1$ to output neuron at layer L, w_i^L ($1 \leqslant i \leqslant M^{L-1}$), as shown in Figure 3.13, we have to find the training rule as Equation (3.47). In this equation, we have to find the derivative of the error function with respect to w_i^L, $(\frac{\partial E^L}{\partial w_i^L})$. In the network, the weight value w_i^L can influence the output of the network only through the input value y^L of output neuron O. Therefore, the derivative of the error function with respect to weight parameter using the chain rule can be written as

$$\begin{aligned}\frac{\partial E^L}{\partial w_i^L} &= \left[\frac{\partial E^L}{\partial y^L}\right]\left[\frac{\partial y^L}{\partial w_i^L}\right] \\ &= \left[\frac{\partial E^L}{\partial y^L}\right]\left[\frac{\partial}{\partial w_i^L}\left(\sum_{k=1}^{M^{L-1}} w_k^L o_k^{L-1} + b^L\right)\right].\end{aligned}$$

The derivatives $\frac{\partial}{\partial w_i}\left(\sum_{k=1}^{M^{L-1}} w_k^L o_k^{L-1} + b^L\right)$ is zero for all terms in the summation except for when $k = i$, hence,

$$\begin{aligned}\frac{\partial E^L}{\partial w_i^L} &= \left[\frac{\partial E^L}{\partial y^L}\right]\left[\frac{\partial}{\partial w_i^L}\left(w_i^L o_i^{L-1} + b^L\right)\right] \\ &= \frac{\partial E^L}{\partial y^L} o_i^{L-1}.\end{aligned} \tag{3.54}$$

From Equation (3.54), the remaining is to compute the derivative of the error with respect to y^L, $(\frac{\partial E^L}{\partial y^L})$. In this case, y^L in network can influence the output of the network only through the output value o^L. Using the chain rule, we can write as

$$\begin{aligned}\frac{\partial E^L}{\partial y^L} &= \left[\frac{\partial E^L}{\partial o^L}\right]\left[\frac{\partial o^L}{\partial y^L}\right] \\ &= \left[\frac{\partial}{\partial o^L}\frac{1}{2}(t - o^L)^2\right]\left[\frac{\partial}{\partial y^L} y^L\right] \\ &= -(t - o^L).\end{aligned} \tag{3.55}$$

Substituting Equation (3.55) into Equation (3.54), we obtain

$$\frac{\partial E^L}{\partial w_i^L} = -(t - o^L) o_i^{L-1}. \tag{3.56}$$

Substituting Equation (3.56) in Equation (3.47), we obtain the training rule of weight parameter w_i^L as

$$\Delta w_i^L = (t - o^L) o_i^{L-1}.$$

To get the update rule of the weight parameter w_i^L, the training rule of w_i^L will be combined with the learning rate. Therefore, we obtain the update rule of weight parameter w_i^L as

$$\Delta w_i^L = \eta_w \Big[(t - o^L) o_i^{L-1} \Big],$$

where

η_w is the learning rate of weight parameter,

t is the target value,

o^L is the output value of output neuron,

o_i^{L-1} is the output value of hidden neuron i at layer $L-1$ ($1 \leqslant i \leq M^{L-1}$).

Case 2: Updating the weight values for hidden neuron

To get the training rule for the weight parameters of hidden neurons, we have to find the derivative of the error function with respect to w_{ij}^l, $(\frac{\partial E^l}{\partial w_{ij}^l})$. There are two cases for updating the weight values of hidden neurons: case 2a is updating the weight values of hidden neurons in layer $3 \leqslant l \leqslant L-1$ and case 2b is updating the weight values of hidden neurons in layer $l = 2$.

Case 2a: Updating the weight values of hidden neurons in layer $3 \leqslant l \leqslant L-1$

In the network, the weight value w_{ij}^l can influence the output of the network only through the input value y_j^l (see Figure 3.14). Therefore, the derivative of the error function with respect to weight parameter using the chain rule can be written

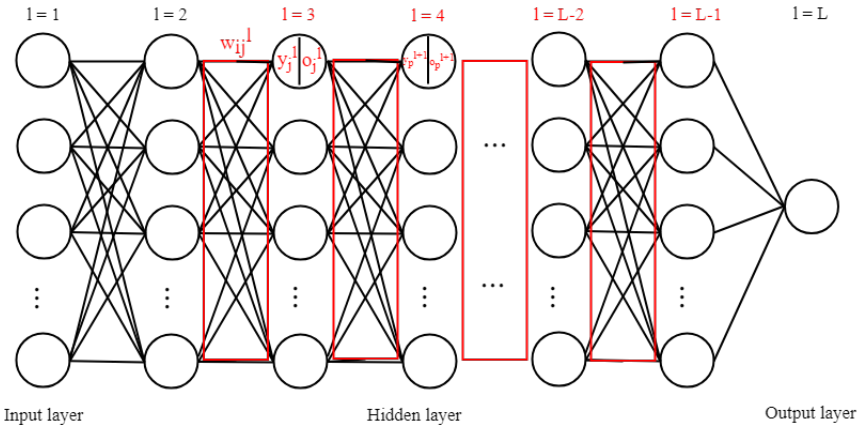

Figure 3.14: Updating the weight values for hidden neuron at layer $3 \leqslant l \leq L-1$

as

$$\begin{aligned}\frac{\partial E^L}{\partial w_{ij}^l} &= \left[\frac{\partial E^L}{\partial y_j^l}\right]\left[\frac{\partial y_j^l}{\partial w_{ij}^l}\right] \\ &= \left[\frac{\partial E^L}{\partial y_j^l}\right]\left[\frac{\partial}{\partial w_{ij}^l}\left(\sum_{k=1}^{M^{l-1}} w_{kj}^l o_k^{l-1} + b_j^l\right)\right].\end{aligned}$$

The derivatives $\frac{\partial}{\partial w_{ij}^l}\left(\sum_{k=1}^{m} w_{kj}^l o_k^{l-1} + b_j^l\right)$ is zero for all terms in the summation except for when $k = i$, hence,

$$\begin{aligned}\frac{\partial E^L}{\partial w_{ij}^l} &= \left[\frac{\partial E^L}{\partial y_j^l}\right]\left[\frac{\partial}{\partial w_{ij}^l}\left(w_{ij}^l o_i^{l-1} + b_j^l\right)\right] \\ &= \frac{\partial E^L}{\partial y_j^l} o_i^{l-1}.\end{aligned} \tag{3.57}$$

From Equation (3.57), the remaining is to compute the derivative of the error with respect to y_j^l, $(\frac{\partial E^L}{\partial y_j^l})$. The input value y_j^l can influence the output of the network through the input value y_p^{l+1}, $(1 \leqslant p \leqslant M^{l+1})$ of hidden neuron in layer

$l+1$. Therefore, we can write in term of the chain rule as

$$\frac{\partial E^L}{\partial y_j^l} = \sum_{p=1}^{M^{l+1}} \left[\frac{\partial E^L}{\partial y_p^{l+1}}\right]\left[\frac{\partial y_p^{l+1}}{\partial y_j^l}\right].$$

The input value y_j^l can influence the output of the network only through the output value o_j^l. Using the chain rule, we can write

$$\begin{aligned}\frac{\partial E^L}{\partial y_j^l} &= \sum_{p=1}^{M^{l+1}} \left[\frac{\partial E^L}{\partial y_p^{l+1}}\right]\left[\frac{\partial y_p^{l+1}}{\partial o_j^l}\frac{\partial o_j^l}{\partial y_j^l}\right] \\ &= \sum_{p=1}^{M^{l+1}} \left[\frac{\partial E^L}{\partial y_p^{l+1}}\right]\left[\frac{\partial}{\partial o_j^l}\left(\sum_{q=1}^{M^l} w_{qp}^{l+1} o_q^l + b_p^{l+1}\right)\frac{\partial}{\partial y_j^l}\left(\frac{1}{1+e^{(-y_j^l)}}\right)\right].\end{aligned}$$

We denote $\dfrac{\partial E^L}{\partial y_p^{l+1}} = -\delta_p^{l+1}$ which can be computed by

$$\delta_p^{l+1} = o_p^{l+1}(1-o_p^{l+1})\sum_{r=1}^{M^{l+2}} w_{pr}^{l+2}\delta_r^{l+2}, (1 \leqslant r \leqslant M^{l+2}).$$

The derivative of $\dfrac{\partial}{\partial o_j^l}\left(\sum\limits_{q=1}^{M^l} w_{qp}^{l+1} o_q^l + b_p^{l+1}\right)$ is zero for all terms in the summation except when $q=j$, hence,

$$\begin{aligned}\frac{\partial E^L}{\partial y_j^l} &= \sum_{p=1}^{M^{l+1}}\left[-\delta_p^{l+1}\right]\left[\frac{\partial}{\partial o_j^l}\left(w_{jp}^{l+1}o_j^l + b_p^{l+1}\right)\right]\left[\frac{\partial}{\partial y_j^l}\left(\frac{1}{1+e^{(-y_j^l)}}\right)\right] \\ &= \sum_{p=1}^{M^{l+1}}\left[-\delta_p^{l+1}\right]\left[w_{jp}^{l+1}\right]\left[\left(\frac{1}{1+e^{(-y_j^l)}}\right)\left(1-\frac{1}{1+e^{(-y_j^l)}}\right)\right] \\ &= \sum_{p=1}^{M^{l+1}}\left[-\delta_p^{l+1}\right]\left[w_{jp}^{l+1}\right]\left[o_j^l(1-o_j^l)\right] \\ &= \left[o_j^l(1-o_j^l)\right]\sum_{p=1}^{M^{l+1}}\left[-\delta_p^{l+1}\right]\left[w_{jp}^{l+1}\right].\end{aligned} \tag{3.58}$$

Substituting Equation (3.58) in Equation (3.57), we obtain

$$\frac{\partial E^L}{\partial w_{ij}^l} = \Big[o_j^l(1-o_j^l)\Big] o_i^{l-1} \sum_{p=1}^{M^{l+1}} \Big[-\delta_p^{l+1}\Big]\Big[w_{jp}^{l+1}\Big]. \tag{3.59}$$

Substituting Equation (3.59) in Equation (3.47)

$$\Delta w_{ij}^l = \Big[o_j^l(1-o_j^l)\Big] o_i^{l-1} \sum_{p=1}^{M^{l+1}} \Big[w_{jp}^{l+1}\Big]\Big[\delta_p^{l+1}\Big]. \tag{3.60}$$

The training rule of weight parameter w_{ij}^l of hidden neuron H_j^l will be combined with the learning rate to obtain the update rule as

$$\Delta w_{ij}^l = \eta_w \Big[o_j^l(1-o_j^l)\Big] o_i^{l-1} \sum_{p=1}^{M^{l+1}} \Big[w_{jp}^{l+1}\Big]\Big[\delta_p^{l+1}\Big], \tag{3.61}$$

where

η_w is the learning rate of weight parameter,

o_j^l is the output value of hidden neuron j at layer l ($1 \leqslant j \leqslant M^l$, $3 \leqslant l \leqslant L-1$),

x_i^{l-1} is the value of input node i at layer $l-1$ ($1 \leqslant i \leqslant M^{l-1}$, $3 \leqslant l \leqslant L-1$),

w_{jp}^{l+1} is the weight value from hidden neuron j at layer l to hidden neuron p at layer $l+1$ l ($1 \leqslant j \leqslant M^l$, $1 \leqslant p \leqslant M^{l+1}$, $3 \leqslant l \leqslant L-1$),

$\delta_p^{l+1} = o_p^{l+1}(1-o_p^{l+1}) \sum_{r=1}^{N^{l+2}} w_{pr}^{l+2}\delta_r^{l+2}$.

Equation (3.61) will first be used to update the weight values at layer $l = L-1$. Next, we will move to update the weight values for layer $l = L-2, l = L-3, ...,$ and $l = 3$ in order.

After update the weight values of hidden neuron at layer $l = 3$, we will move to update the weight values of hidden neurons at layer $l = 2$ in the next step.

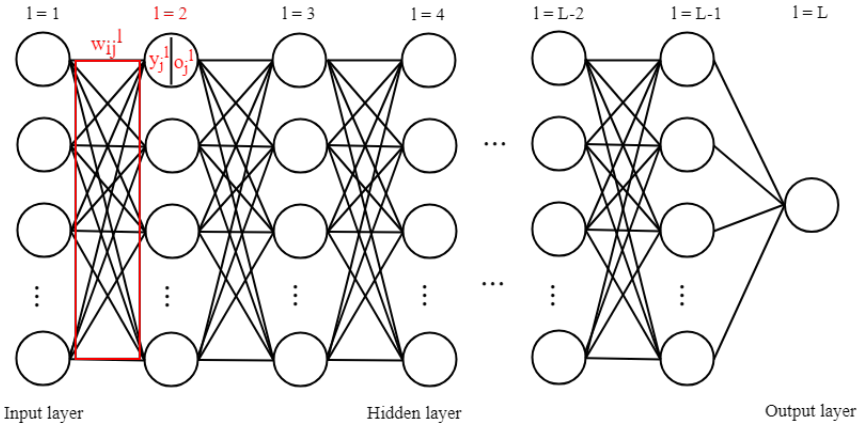

Figure 3.15: Updating the weight values for hidden neuron at layer $l = 2$

Case 2b: Updating the weight values of hidden neuron in layer $l = 2$

To get the update rule of weight parameter in layer $l = 2$, we can derive the formula similar to Case 2a. Therefore, we obtain the update rule as

$$\Delta w_{ij}^2 = \eta_w \left[o_j^2(1 - o_j^2) \right] x_i \sum_{p=1}^{M^3} \left[w_{jp}^3 \right] \left[\delta_p^3 \right], \tag{3.62}$$

where

η_w is the learning rate of weight parameter,

o_j^2 is the output value of hidden neuron j at layer l $(1 \leqslant j \leqslant M^l)$,

x_i is the value of input node i at layer 1 $(1 \leqslant i \leqslant M^1)$,

w_{jp}^3 is the weight value from hidden neuron j at layer 2 to hidden neuron p at layer 3 $(1 \leqslant j \leqslant M^2,\ 1 \leqslant p \leqslant M^3)$,

$\delta_p^{l+1} = o_p^{l+1}(1 - o_p^{l+1}) \sum_{r=1}^{M^{l+2}} w_{pr}^{l+2} \delta_r^{l+2}$.

Equation (3.62) will be used to update the weight values in layer $l = 2$. From the Step 1 as forward propagation to the Step 2 as backward propagation, we can update all weight values of a feedforward neural network. Next, we will repeat the same process using additional input data until we obtain acceptable error or until we reach some predefined number of updating.

3.8 Deep belief networks

Deep belief network (DBN) is stacked of restricted Boltzmann machines, the structure is shown as Figure 3.16. As we know that restricted Boltzmann machine consists of two layers, when two restricted Boltzmann machines stack together, it means that the hidden layer of the first restricted Boltzmann machine becomes visible layer of the second restricted Boltzmann machine. In this manner, deep belief network can be formed by stacking of many restricted Boltzmann machines.

3.9 Learning deep belief networks

The target of learning a deep belief network is to find the appropriate weight values between each layer (see Figure 3.16). To learn the weight values of a deep belief network encompass two paths, unsupervised path and supervised path. In unsupervised path, we will learn each restricted Boltzmann machine in order to use the minimizing contrastive divergence algorithm to obtain the weight values for all connections in the network. The final weight values from unsupervised path will be used to initialize wight values in supervised path. To train the deep belief

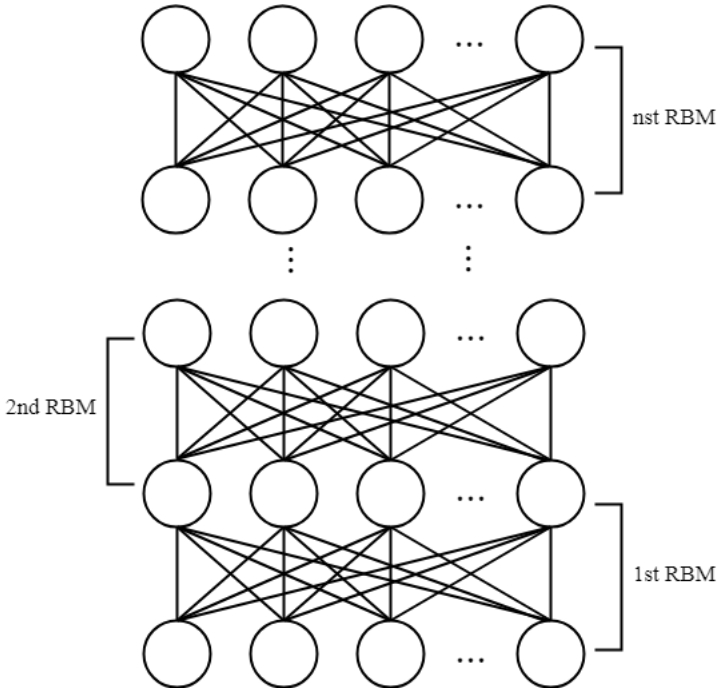

Figure 3.16: Deep belief network with n restricted Boltzmann machines stack together

network in supervised path, we will use the back-propagation neuron network algorithm to obtain the weight values which minimize the error function.

Chapter4

RESULT AND DISCUSSION

The main objective in this research is to use a deep belief network with restricted Boltzmann machines to predict drought. Since there is no mature method to determine architecture of a deep belief network, the experimental method will be used. Only one network structure which obtains the smallest error will be used to predict drought. Several factors cause drought, they are land, water temperature, air circulation, and soil moisture. The main factor that causes drought in Nakhon Ratchasima province is the lack of rain. Therefore, the rainfall data will be considered which the detail is explained in Section 4.1. These rainfall data will be transformed into SPI, the process to transform is shown in Section 4.2. There are three criteria to evaluate the models, these criteria are explained in Section 4.3. The process to predict drought using a deep belief network is illustrated in Section 4.4. The results from Section 4.4 will be shown in Section 4.5.

4.1 Data

Nakhon Ratchasima, a northeastern province of Thailand, has a total area of 12.80 million hectares, out of which 8.70 million hectares (representing 67.97%)

are agricultural land. The main occupation of the inhabitants in the province is farming and the majority of those farms produce rice, corn, and cassava. Due to nature of their work, availability of water is of much importance. They are dependent more on natural rainfall for their crop to survive.

Forecasting the availability or non-availability of rainfall is very important to ensure rice production in the province. One of the main problems faced by the many farmers in Nakhon Ratchasima province is seasonal (or perhaps annual) shortage of rain, causing severe drought, and affecting rice production. As a result, this research focuses on predicting drought by using rainfall data in Nakhon Ratchasima province.

Table 4.1: Rainfall data (millimeters)

Year	**1957**	**1958**	**1959**	**1960**	**1961**	**1962**	**1963**	**...**	**2015**
Jan.	0	0	0	0	0	0	0	...	16.3
Feb.	10.5	24.1	0	4.9	0	0	0	...	0
Mar.	0.9	17.6	28.8	69.6	8.3	77.5	15.2	...	0
Apr.	65.4	47.6	63.4	37.6	64.0	35.5	145.8	...	50.8
May.	68.1	87.1	92.0	62.1	220.3	128.0	80.7	...	85.5
June.	58.9	83.1	83.8	100.9	68.1	84.9	99.2	...	56.2
July.	214.9	128.2	170.2	65.9	62.6	131.4	89.3	...	156.0
Aug.	50.0	157.5	97.1	81.2	40.4	150.0	129.4	...	208.4
Sep.	222.8	377.6	515.9	240.2	155.5	367.6	281.5	...	237.1
Oct.	194.4	164.2	188.7	4	142.0	199.9	162	...	86.3
Nov.	5.8	0.8	2.6	0	0	59.2	70.8	...	3
Dec.	0	0	0	0	0	0	0	...	0.3

Data for this research is rainfall data, measured by Hydrology and water management center, lower northeastern region, Muang district, Nakhon Ratchasima province. The example of rainfall data is shown in Table 4.1. The data consist of 708 monthly rainfall observations from January 1957 to December 2015, each

observation is the mean value of daily rainfall measured in millimeters. It is the secondary data obtained from the website of the Bureau of Water Management and Hydrology (Centre for Hydrology Irrigation northeastern Lower, 2016). These rainfall data are recorded by non-recording rain gauge which use cylinder in unit of millimeters to measure.

The minimum and maximum of rainfall data are 0 and 515.9 millimeters, respectively. The mean value is using to measure the center of the data which calculate from sum of observations divided by the number of observations, the mean value of rainfall data is 85.98 millimeters. The standard deviation is 86.75 millimeters, which explains the dispersion of rainfall data.

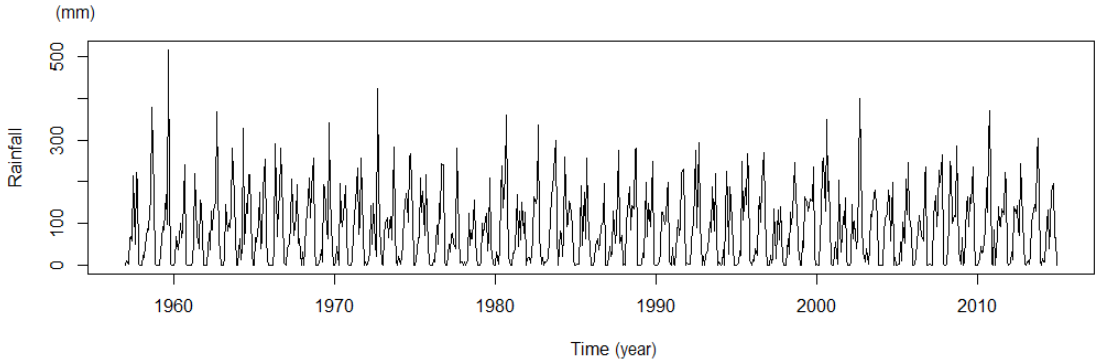

Figure 4.1: Rainfall data from 1959-2015 (millimeters)

Figure 4.1 shows the graph of rainfall data. The x-axis represents year and y-axis represents rainfall data in unit millimeters. These rainfall data contain seasonal pattern.

Figure 4.2 shows the histogram of rainfall data. The x-axis represents rainfall data and y-axis represents the density or probability. The rainfall data have been split into bins, each bin represents a 10 millimeters. The probability in

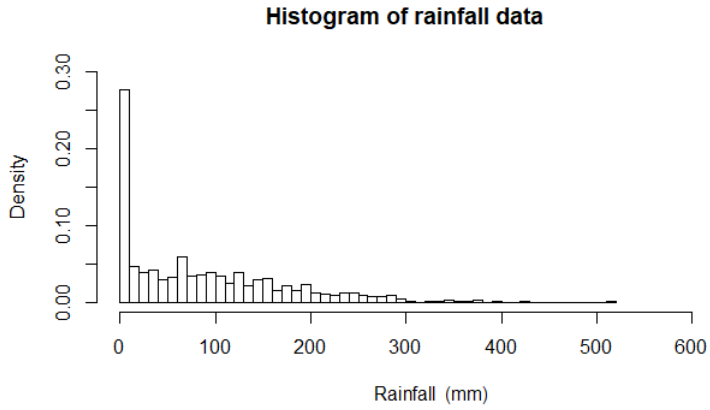

Figure 4.2: Histogram of rainfall data, each bin represents 10 millimeters

y-axis can be computed as frequency of each bin (10 millimeters) divided with all number of bins.

4.2 Computing standardized precipitation index

Standardized precipitation index (SPI) is a drought index. There are many difference time-scales which SPI can be calculated from 1 month to 72 months, denoted SPI1, SPI2, SPI3, ..., SPI72, respectively. For example, SPI3 is computing drought in period of three months which compares with the long term of rainfall data in the same months on recorded. SPI6 is computing drought in period of six months which compares with the long term of rainfall data in the same months on recorded. It is the same as other time-scale of SPI which computing drought in period of its months compares with the long term of rainfall data in the same months on recored. Statistically, the best result of standardized precipitation index ranges from 1-24 months (Guttman, 1999).

To transform rainfall data into SPI, we will focus on SPI3, SPI6, SPI9, and SPI12. The purpose of transforming rainfall data into SPI index is to transform the

distribution of rainfall data into normal distribution with mean zero and standard deviation of one. The process to transform rainfall data into SPI3 will be shown in Table 4.2 and explained as follows.

(1) Calculating the average of monthly rainfall data from January to

Table 4.2: The rainfall data from January-March, 1957-2015

Year	**Month**	**Rainfall (millimeters)**	**SPI3**
1957	Jan.	0	
	Feb.	10.5	
	Mar.	0.9	
	x_{1957}	3.8	-1
1958	Jan.	0	
	Feb.	24.1	
	Mar.	17.6	
	x_{1958}	13.9	-0.9
1959	Jan.	0	
	Feb.	0	
	Mar.	28.8	
	x_{1959}	9.6	-0.47
1960	Jan.	0	
	Feb.	4.9	
	Mar.	69.6	
	x_{1960}	24.83	0.61
⋮	⋮	⋮	⋮
2015	Jan.	16.3	
	Feb.	0	
	Mar.	0	
	x_{2015}	5.43	-0.98

March of each year. Let x_t denote the average rainfall data from January to March in year t. For example, the average of rainfall from January to March in 1957 is 3.8 (as seen in Table 4.2). The formula to calculate the average is

$$x_t = \sum_{i=1}^{3} \frac{x_i}{3}$$

where

x_i is the value of observation in time-scale.

(2) Calculating $\bar{x}$ which is the average of $x_{1957}, x_{1958}, x_{1959}, ..., x_{2015}$. In this data, $\bar{x} = 18.9$.

(3) Estimating the shape parameter $(\hat{\gamma})$, $\hat{\gamma} = 1.1143$ which can be computed as

$$\hat{\gamma} = \frac{1 + \sqrt{1 + \frac{4A}{3}}}{4A},$$

where

$$A = ln\bar{x} - \frac{1}{59} \sum_{i=1957}^{2015} lnx_i,$$

(4) Estimating the scale parameter $(\hat{\beta})$, $\hat{\beta} = 16.9740$ which can be computed as

$$\hat{\beta} = \frac{\bar{x}}{\hat{\gamma}}.$$

(5) Testing the $\gamma = \hat{\gamma}$ and $\beta = \hat{\beta}$ achieve the maximum value

$$\begin{aligned} \mathbb{D} &= \left[f_{\beta\beta}(\hat{\gamma}, \hat{\beta})\right] \left[f_{\gamma\gamma}(\hat{\gamma}, \hat{\beta})\right] - \left[f_{\beta\gamma}(\hat{\gamma}, \hat{\beta})\right]^2 \\ &= (-0.2282)(-62.3870) - (-3.4759)^2 \\ &= 2.1548 \end{aligned}$$

which $\mathbb{D} > 0$ and $f_{\beta\beta}(1.1143, 16.9740) < 0$, $f(1.1143, 16.9740)$ is maximum.

(6) Using $\hat{\gamma} = 1.1143$ and $\hat{\beta} = 16.9740$ to calculate the cumulative proba-

bility density function ($H(x)$), $H(x) = 0.4491$ which can be computed as

$$H(x) = u + (1 - u)F(x),$$

where

u is probability of zero precipitation,

$F(x) = \frac{1}{\Gamma(\hat{\gamma})} \int_0^x t^{\hat{\gamma}-1} e^{-t} dt,$

$t = \frac{x}{\hat{\beta}},$

$\Gamma(\hat{\gamma})$ if the gamma function.

(7) Transforming the cumulative probability density function $H(x)$ to be the standard normal random variable Z, known as SPI value. For example, SPI3 for January to March is -1.00 (as seen in Table 4.2). The equation to compute SPI is

$$SPI(x_i) = \begin{cases} -k, & 0 < H(x_i) \leq 0.5 \\ +k, & 0.5 < H(x_i) < 1, \end{cases}$$

where

$$k = p - \frac{c_0 + c_1 p + c_2 p^2}{1 + d_1 p + d_2 p^2 + d_3 p^3},$$

constants c and d and variable p can be calculated as follows:

$$p = \begin{cases} \sqrt{ln\left(\frac{1}{H^2(x)}\right)}, & 0 < H(x_i) \leq 0.5 \\ \sqrt{ln\left(\frac{1}{(1-H(x))^2}\right)}, & 0.5 < H(x_i) < 1, \end{cases}$$

$$c_0 = 2.515517, c_1 = 0.802853, c_2 = 0.010328,$$

$$d_1 = 1.432788, d_2 = 0.189269, d_3 = 0.001308.$$

From the process (1) to the process (7), we can get the first value of SPI3 which is -1.00. This value means that in 1957, January-March moderately dry when the value compares with January-March of every year on recorded.

To get the second value of SPI3, we will use the data from February-April of 1957-2015. Calculating from the process (1) to the process (7), we can get the second value of SPI3 which is -0.44. This value means that February-April in 1957 near normal when the value compares with February-April of every years on recorded. These processes will be repeated until we use all rainfall data.

4.3 Evaluation matrices

To evaluate the performance of our models, three criteria will be considered. They are root mean square error (RMSE), mean absolute error (MAE), and R-squared. RMSE and MAE are the functions to calculate the error of the model. R-squared is the statistical measure of how close the observations are to the fitted regression line. The value of R-squared range for 0 to 1. If R-squared is 1, all observations fall perfectly on the regression line and if R-squared is 0, the estimated

regression line is perfectly horizontal. The equation to compute these criteria are shown as follows

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(t_i - o_i)^2}{n}}, \tag{4.1}$$

$$\text{MAE} = \frac{\sum_{i=1}^{n}|t_i - o_i|}{n}, \tag{4.2}$$

$$\text{R-squared} = 1 - \frac{\sum_{i=1}^{n}(t_i - o_i)^2}{\sum_{i=1}^{n}(t_i - \bar{t})^2}, \tag{4.3}$$

where

t_i is the target value of $SPI(x_i)$,

o_i is the output value from model,

n is the total number of SPI data,

$\bar{t}$ is the mean value of SPI.

4.4 Experimental Methodology

The process to predict drought is described in flowchart displayed in Figure 4.3. The beginning of the process is obtaining the data. This data will be transformed into standardized precipitation index (SPI) in step 1, then we obtain SPI data. The detail about SPI data obtaining in step 1 is explained in Section 4.4.4. Step 2 is to separate data into three groups (1) Structure Training Set which contains $x_1, x_2, x_3, ..., x_m$, they are 80% of the data (2) Weight Training Set which contains $x_1, x_2, x_3, ..., x_{p-1}$ where $m + 1 \leqslant p \leqslant n$ and (3) Test Data which is x_p. The detail about how to generate those data sets is explained in Section 4.4.2.

Step 3 is to determine suitable network architecture for our problem. In this step, we will use data from Structure Training Set to train and evaluate various network architectures and select the one with the least error. Let DBN(A) be the network structure with the least error. The detail on how to train, evaluate, and select such structure is described in Section 4.4.3. Step 4, we will take DBN(A) from step 3 which contains no information about the weights of the network and train this network using data from Weight Training Set. This training includes both unsupervised and supervised learning. The result of step 4 is DBN(B) which is a deep belief network that will be used in step 5 to predict Test Data x_p and obtain an accuracy.

4.4.1 Result of transformation of SPI

To predict drought, rainfall data has to be transformed into standardized precipitation index. In this research, we only focus on SPI3, SPI6, SPI9, and SPI12. The rainfall data will be transformed into SPI with difference time-scale using spi_sl_6 software from world meteorological organization (WMO).

When the rainfall data is transformed into standardized precipitation index of different time-scale, the number of observations is reduced by $s - 1$ when s is time-scale. For example of SPI3, the number of observation will be reduced by 2. Therefore, the number of observation reduced from 708 to 706 observations.

Table 4.3 shows the total observation after transforming the rainfall data into drought index of difference time-scale. The remaining of observations for SPI3,

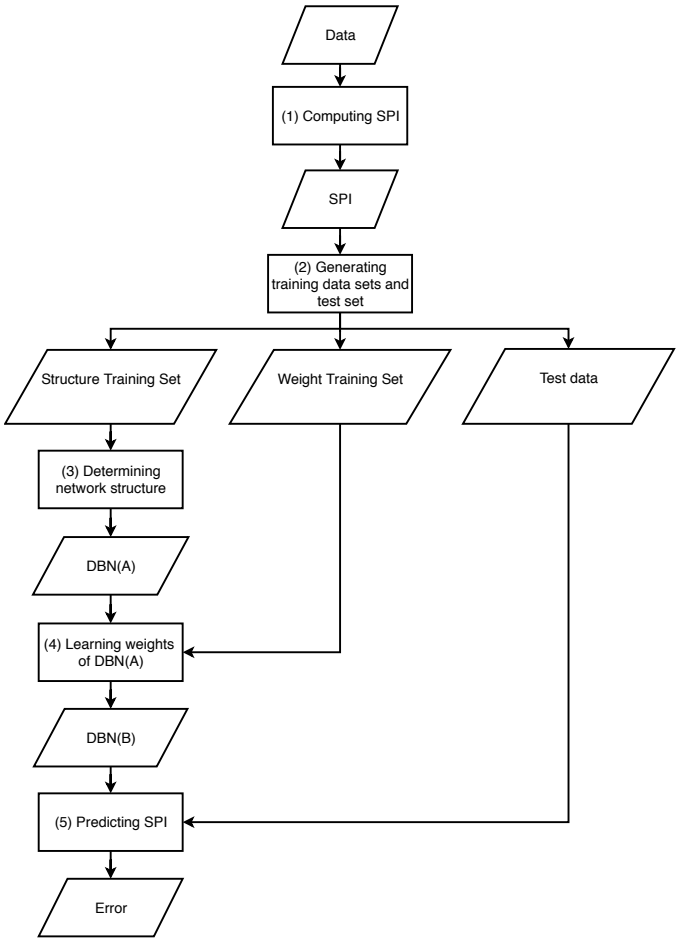

Figure 4.3: The research framework to predict SPI

Table 4.3: The total observations after transformed into standardized precipitation index

SPI	**Total observation**	**80% of the data**	**20% of the data**
SPI3	706	564	142
SPI6	703	562	141
SPI9	700	560	140
SPI12	697	557	140

SPI6, SPI9, and SPI12 are 706, 703, 700, and 697 observations, respectively. The number of 80% and 20% of the data will be used in walk forward validation which will be explained in Section 4.4.2.

4.4.2 Generating training data sets and test set

The target of predicting time series data is to make accurate prediction in the future. There are many difference ways to evaluate the model. In this research, we will split data into two sets, training set and test set. There are many methods to split data used for training and testing such as train-test splits and k-fold cross validation, but these methods are not suitable for time series data because they disregard the temporal components which commonly exists in this type of data. In this research, we employ walk forward validation to evaluate the performance of our model. Traditionally, this method will split data into two groups, they are training set and test set, the period to split depends on users. Let $x_1, x_2, x_3, ..., x_n$ be time series data consist of n observations. Suppose that we are using m records for training and the remaining $n-m$ records for testing. For example of SPI3, the training data are $x_1, x_2, x_3, ..., x_{564}$ and the test data are $x_{565}, x_{566}, x_{567}, ..., x_{706}$. In the first step, the data in training set will be used to train and find the best network to predict the value of x_{565}. Next, $x_1, x_2, x_3, ..., x_{564}, x_{565}$ will be used to train and find the best network to predict the value of x_{566}. It will perform in this manner until we are able to predict the value of all data in the test set.

Table 4.4: Training sets and test set

Structure training set	**Weight training**	**Test data**
$x_1, x_2, x_3, ..., x_m$	$x_1, x_2, x_3, ..., x_{p-1}$ where $x_p, m+1 \leqslant p \leqslant n$	x_p

However, determining a network structure in this research is very time consuming. Therefore, we will use a modified version of walk forward validation. Beginning, we will split data into three groups, (1) Structure Training Set, (2) Weight Training, and (3) Test data, which is shown in Table 4.4. To use modified version of walk forward validation, in the first step, we used the data 80%, the data is $x_1, x_2, x_3, ..., x_m$ to train and find the best network structure. Then we will keep this network structure fixed and use $x_1, x_2, x_3, ..., x_{p-1}$, where $m+1 \leqslant p \leqslant n$ to learn a new set of weight values for our deep belief network and use this network to predict the value of x_p.

4.4.3 Determining network structure

There is no mature method to determine architecture of deep belief networks. Therefore, to create architecture of a deep belief network, we have to decide how many restricted Boltzmann machines will be stacked together. Furthermore, we have to decide how many neurons in visible layer and how many neurons in hidden layer for the first restricted Boltzmann machine. When we stack second restricted Boltzmann machine, it means that the hidden layer of the first restricted Boltzmann machine will be visible layer for the second restricted Boltzmann machine. In this manner, we can create the structure of a deep belief network by stacking of n restricted Boltzmann machine together.

In this research, the architecture of a deep belief network consists of two restricted Boltzmann machines stacked together shown in Figure 4.4. This research used experimental method by trying all combinations of network structures con-

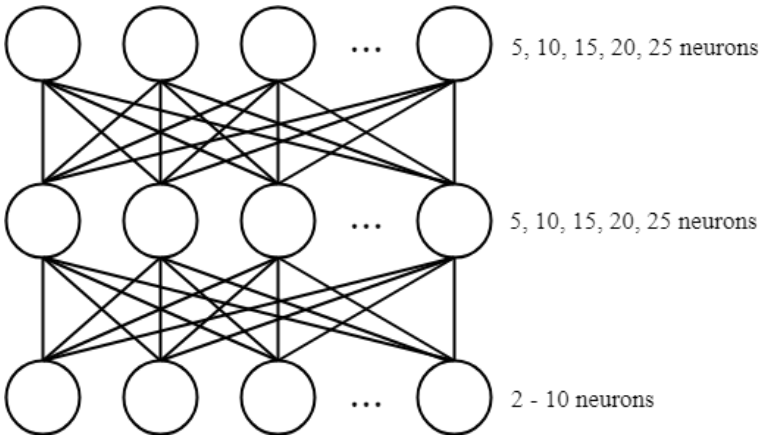

Figure 4.4: Stacking of 2 restricted Boltzmann machines

sisting of 2 to 10 neurons in visible layer and 5, 10, 15, 20, and 25 neurons in hidden layer. There are the total of 45 network structures in which the best one with minimum error will be chosen. The chosen network structure will be used to learn the weight values in unsupervised path and supervised path. These 45 network structures will be chosen one network structure which obtains the smallest error. This structure will be fixed and we will add the second restricted Boltzmann machine in which the hidden layer of the first restricted Boltzmann machine becomes visible layer of the second restricted Boltzmann machine. There are 5, 10, 15, 20, and 25 neurons in hidden layer for second restricted Boltzmann machine. To learn the weight values, the weight values of the first restricted Boltzmann machine will be fixed. Then we will learn the weight values for second restricted Boltzmann machine using unsupervised path and supervised path. These 5 network structures will be chosen only one best structure which contains the smallest error.

To predict drought, the best network structure without the weight values will be used to learn the weight values using deep belief network. Learning

deep belief network consists of two paths, unsupervised path and supervised path. In unsupervised path, we will learn the weight values using continuous restricted Boltzmann machine with MCD algorithm. In supervised path, we will consider the best network structure as the feedforward neuron network structure (see Section 3.7). In this path, we use back-propagation algorithm to learn the weight values. The best network structure through learning the weight values by unsupervised path and supervised path will be used to predict drought.

4.5 Experimental results

There are two experimental results. The first is the best network structure explained in Section 4.5.1. This section shows the experimental result of SPI3, SPI6, SPI9, and SPI12. The different time-scale of SPI will be chosen one best network structure which contains the smallest error. The second is prediction, the detail is shown in Section 4.5.2.

4.5.1 The best network structure

There are four different time-scales of standardized precipitation index. They are SPI3, SPI6, SPI9, and SPI12 which means that we have four data sets of difference time-scales. Therefore, we have to find the best network structure for each of those data sets. The experimentation result of each difference time-scale is shown as follows.

The experimental result of SPI3

Table 4.5 shows the results of input layer with first hidden layer of SPI3. When we compared the RMSE of two input nodes to ten input nodes, the results showed that the smallest average error is 0.7074 which occurs at 3 nodes. Next, focus was placed on the smallest error. The result shows that the first hidden layer with 25 neurons had the smallest RMSE, which RMSE is 0.6984. Hence, the best network structure with first hidden layer is 3-25-1 (3 input nodes - 25 neurons for first hidden layer - 1 output neuron). This structure will be fixed for adding the second hidden layer in the next step.

In experimentation to add a second hidden layer, the number of nodes in input layer and the number of neurons in hidden layer were taken from the previous process. It is 3 nodes in input layer and 25 neurons in the first hidden layer. This structure was added to the second hidden layer with experimentation of 5, 10, 15, 20, and 25 neurons. The effect is displayed in Table 4.6. The best structure occurred at 15 neurons in the second hidden layer, the smallest RMSE is 0.6925. Therefore, the best network structure of SPI3 in this experimentation is 3-25-15-1 (3 input nodes - 25 neurons in the first hidden layer - 15 neurons in the second hidden layer - 1 output neuron).

The experimental result of SPI6

The experimentation results of SPI6 by adding the first hidden layer are shown in the Table 4.7. We found that the smallest RMSE occurs at 8 nodes in input layer and 20 neurons in the first hidden layer. Therefore, the best structure

with adding the first hidden layer is 8-20-1 (8 input nodes - 20 neurons for the first hidden layer - 1 output neuron) which the value of RMSE is 0.4374. This structure will be fixed for adding second hidden layer.

The results for adding the second hidden layer are shown in the Table 4.8. The smallest RMSE is 0.4256 which occurs at 25 neurons in the second hidden layer. Therefore, the best network structure of SPI6 is 8-20-25-1 (8 input nodes - 20 neurons for the first hidden layer - 25 neurons for the second hidden layer - 1 output neuron).

The experimental result of SPI9

Table 4.9 illustrated the results of experimentation by adding the first hidden layer for SPI9. The result shows that the smallest RMSE is 0.4175 which occurs at 10 nodes in input layer and 15 neurons for the first hidden layer. This structure will be fixed for adding the second hidden layer.

The experimentation results by adding the second hidden layer are shown in the Table 4.10. In comparison the RMSE, we found that 20 neurons for the second hidden layer has smallest RMSE. Therefore, the best network structure of SPI6 is 10-15-20-1 (10 input nodes - 15 neurons for the first hidden layer - 20 neurons for the second hidden layer - 1 output neuron).

The experimental result of SPI12

The experimentation results of the first hidden layer of SPI12 are shown in the Table 4.11. The smallest RMSE is 0.3399 which occurs at 4 nodes in input

layer and 15 neurons in the first hidden layer.

Table 4.12 shows the results for adding the second hidden layer. The smallest RMSE in comparison of surrounding neurons is 0.3388 which occurs at 10 neurons in the second hidden layer. Therefore, the best network structure of SPI12 by adding the second hidden layer is 4-15-10-1 (4 input nodes - 15 neurons for the first hidden layer - 10 neurons for the second hidden layer - 1 output neuron).

Summary of the best network structure of each SPI are shown in Table 4.13.

Table 4.5: Effects of input layer with first hidden layer of SPI3

Input	**Hidden**	**RMSE**	**MAE**	**Input**	**Hidden**	**RMSE**	**MAE**
2	5	0.7317	0.5970	7	5	0.7421	0.6136
2	10	0.7351	0.5987	7	10	0.7168	0.5895
2	15	0.7232	0.5864	7	15	0.7391	0.6091
2	20	0.7168	0.5788	7	20	0.7071	0.5795
2	25	0.7261	0.5897	7	25	0.7327	0.6019
	Avg	0.7265	0.5901		Avg	0.7275	0.5987
3	5	0.7344	0.6037	8	5	0.7051	0.5799
3	10	0.6997	0.5662	8	10	0.7571	0.6224
3	15	0.7039	0.5715	8	15	0.7275	0.6016
3	20	0.7006	0.5693	8	20	0.6966	0.5728
3	**25**	**0.6984**	**0.5649**	8	25	0.7009	0.5741
	Avg	**0.7074**	**0.5751**		Avg	0.7174	0.5901
4	5	0.7053	0.5736	9	5	0.7571	0.6270
4	10	0.7179	0.5857	9	10	0.7078	0.5838
4	15	0.7320	0.5991	9	15	0.7282	0.6017
4	20	0.7162	0.5851	9	20	0.7247	0.5962
4	25	0.7132	0.5808	9	25	0.6997	0.5769
	Avg	0.7169	0.5848		Avg	0.7235	0.5971
5	5	0.7350	0.6024	10	5	0.7412	0.6081
5	10	0.7013	0.5678	10	10	0.7049	0.5784
5	15	0.7254	0.5934	10	15	0.7218	0.5979
5	20	0.6916	0.5599	10	20	0.7379	0.6088
5	25	0.7055	0.5713	10	25	0.7102	0.5842
	Avg	0.7117	0.5789		Avg	0.7232	0.5954
6	5	0.7565	0.6247				
6	10	0.7370	0.6068				
6	15	0.7147	0.5861				
6	20	0.7142	0.5829				
6	25	0.7050	0.5737				
	Avg	0.7254	0.5948				

Table 4.6: Effects of second hidden layer of SPI3

Input	**Hidden1**	**hidden2**	**RMSE**	**MAE**
		5	0.9537	0.7621
		10	0.7459	0.6117
3	**25**	**15**	**0.6925**	**0.5604**
		20	0.7162	0.5850
		25	0.7172	0.5867
		Avg	0.7651	0.6211

Table 4.7: Effects of input layer with first hidden layer of SPI6

Input	**Hidden**	**RMSE**	**MAE**	**Input**	**Hidden**	**RMSE**	**MAE**
2	5	0.5187	0.4083	7	5	0.4622	0.3459
2	10	0.4883	0.3807	7	10	0.4466	0.3333
2	15	0.4863	0.3791	7	15	0.4355	0.3207
2	20	0.4616	0.3545	7	20	0.4474	0.3343
2	25	0.4891	0.3819	7	25	0.4357	0.3219
	Avg	0.4888	0.3809		Avg	0.4454	0.3312
3	5	0.4974	0.3896	8	5	0.4487	0.3373
3	10	0.4828	0.3756	8	10	0.4379	0.3211
3	15	0.4851	0.3757	8	15	0.4434	0.3305
3	20	0.4654	0.3585	**8**	**20**	**0.4374**	**0.3218**
3	25	0.4796	0.3703	8	25	0.4394	0.3260
	Avg	0.4820	0.3739		**Avg**	**0.4413**	**0.3273**
4	5	0.4616	0.3572	9	5	0.4660	0.3513
4	10	0.4668	0.3607	9	10	0.4730	0.3598
4	15	0.4586	0.3541	9	15	0.4522	0.3353
4	20	0.4606	0.3550	9	20	0.4511	0.3363
4	25	0.4633	0.3600	9	25	0.4399	0.3217
	Avg	0.4621	0.3574		Avg	0.4564	0.3408
5	5	0.4449	0.3390	10	5	0.4404	0.3261
5	10	0.4661	0.3585	10	10	0.4451	0.3271
5	15	0.4526	0.3473	10	15	0.4513	0.3351
5	20	0.4444	0.3388	10	20	0.4282	0.3108
5	25	0.4524	0.3470	10	25	0.4439	0.3281
	Avg	0.4520	0.3461		Avg	0.4417	0.3254
6	5	0.4618	0.3437				
6	10	0.4426	0.3242				
6	15	0.4486	0.3296				
6	20	0.4436	0.3261				
6	25	0.4483	0.3280				
	Avg	0.4489	0.3303				

Table 4.8: Effects of second hidden layer of SPI6

Input	**Hidden1**	**hidden2**	**RMSE**	**MAE**
		5	0.9321	0.7374
		10	0.4836	0.3704
8	**20**	15	0.4853	0.3725
		20	0.4397	0.3315
		25	**0.4256**	**0.3142**
		Avg	0.5532	0.4252

Table 4.9: Effects of input layer with first hidden layer of SPI9

Input	**Hidden**	**RMSE**	**MAE**	**Input**	**Hidden**	**RMSE**	**MAE**
2	5	0.4469	0.3362	7	5	0.4342	0.3324
2	10	0.4284	0.3171	7	10	0.4659	0.3614
2	15	0.4446	0.3337	7	15	0.4569	0.3497
2	20	0.4502	0.3384	7	20	0.4558	0.3494
2	25	0.4370	0.3256	7	25	0.4450	0.3400
	Avg	0.4414	0.3302		Avg	0.4515	0.3465
3	5	0.4299	0.3234	8	5	0.4629	0.3538
3	10	0.4459	0.3377	8	10	0.4348	0.3276
3	15	0.4317	0.3212	8	15	0.4350	0.3274
3	20	0.4369	0.3285	8	20	0.4444	0.3357
3	25	0.4604	0.3514	8	25	0.4480	0.3390
	Avg	0.4409	0.3324		Avg	0.4450	0.3367
4	5	0.4336	0.3245	9	5	0.4441	0.3383
4	10	0.4390	0.3279	9	10	0.4397	0.3323
4	15	0.4219	0.3083	9	15	0.4332	0.3286
4	20	0.4389	0.3276	9	20	0.4600	0.3532
4	25	0.4421	0.3308	9	25	0.4403	0.3339
	Avg	0.4351	0.3238		Avg	0.4434	0.3372
5	5	0.4319	0.3228	10	5	0.4330	0.3301
5	10	0.4346	0.3217	10	10	0.4337	0.3282
5	15	0.4491	0.3361	**10**	**15**	**0.4175**	**0.3154**
5	20	0.4418	0.3274	10	20	0.4427	0.3380
5	25	0.4426	0.3287	10	25	0.4283	0.3221
	Avg	0.4400	0.3273		**Avg**	**0.4310**	**0.3267**
6	5	0.4400	0.3412				
6	10	0.4375	0.3335				
6	15	0.4405	0.3321				
6	20	0.4271	0.3220				
6	25	0.4323	0.3269				
	Avg	0.4354	0.3311				

Table 4.10: Effects of second hidden layer of SPI9

Input	**Hidden1**	**hidden2**	**RMSE**	**MAE**
		5	0.5162	0.4057
		10	0.4483	0.3440
10	**15**	15	0.4395	0.3370
		20	**0.4174**	**0.3143**
		25	0.4439	0.3370
		Avg	0.4530	0.3476

Table 4.11: Effects of input layer with first hidden layer of SPI12

Input	**Hidden**	**RMSE**	**MAE**	**Input**	**Hidden**	**RMSE**	**MAE**
2	5	0.3623	0.2630	7	5	0.3774	0.2801
2	10	0.3643	0.2546	7	10	0.3837	0.2849
2	15	0.3497	0.2499	7	15	0.3755	0.2773
2	20	0.3698	0.2716	7	20	0.3770	0.2782
2	25	0.3539	0.2559	7	25	0.3868	0.2883
	Avg	0.3600	0.2590		Avg	0.3800	0.2817
3	5	0.4023	0.3061	8	5	0.3750	0.2744
3	10	0.3687	0.2703	8	10	0.3747	0.2735
3	15	0.3515	0.2541	8	15	0.3907	0.2879
3	20	0.3588	0.2625	8	20	0.3737	0.2746
3	25	0.3739	0.2772	8	25	0.3871	0.2835
	Avg	0.3710	0.2740		Avg	0.3802	0.2787
4	5	0.3579	0.2624	9	5	0.4068	0.3037
4	10	0.3802	0.2834	9	10	0.3944	0.2896
4	**15**	**0.3399**	**0.2463**	9	15	0.3761	0.2763
4	20	0.3410	0.2470	9	20	0.3774	0.2785
4	25	0.3531	0.2600	9	25	0.3712	0.2707
	Avg	**0.3544**	**0.2598**		Avg	0.3851	0.2837
5	5	0.3434	0.2504	10	5	0.3897	0.2894
5	10	0.3582	0.2683	10	10	0.3858	0.2874
5	15	0.3574	0.2629	10	15	0.3914	0.2881
5	20	0.3668	0.2781	10	20	0.3694	0.2716
5	25	0.3629	0.2694	10	25	0.3994	0.2951
	Avg	0.3577	0.2658		Avg	0.3871	0.2863
6	5	0.3666	0.2720				
6	10	0.3623	0.2676				
6	15	0.3831	0.2891				
6	20	0.3799	0.2853				
6	25	0.3823	0.2880				
	Avg	0.3748	0.2804				

Table 4.12: Effects of second hidden layer of SPI12

Input	**Hidden1**	**hidden2**	**RMSE**	**MAE**
		5	0.3716	0.3731
		10	**0.3388**	**0.2398**
4	**15**	15	0.3569	0.2606
		25	0.3679	0.2681
		Avg	0.3603	0.28170

Table 4.13: The best network structure of each SPI

SPI	Best structure	RMSE	MAE
SPI3	3-25-15-1	0.6925	0.5604
SPI6	6-20-25-1	0.4256	0.3142
SPI9	10-15-20-1	0.4174	0.3143
SPI12	4-15-10-1	0.3388	0.2398

4.5.2 Prediction

Figure 4.5 shows the graphs between the target values and predict values of different time-scale. There are four graphs which are SPI3, SPI6, SPI9, and SPI12. For all the graphs, x-axis represents year and y-axis represents SPI. Solid lines represent the target values and dashed lines represents the predict values.

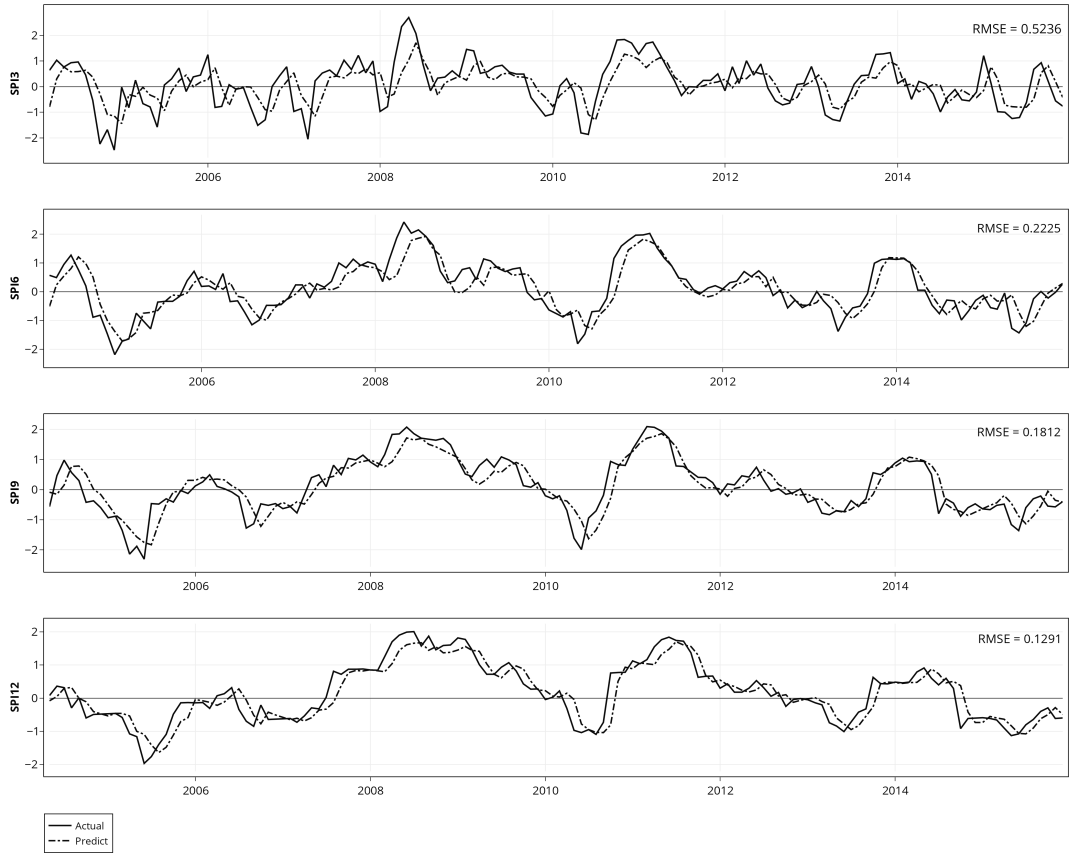

Figure 4.5: Comparison between target values and predict values of SPI which x-axis represents year and y-axis represents SPI value

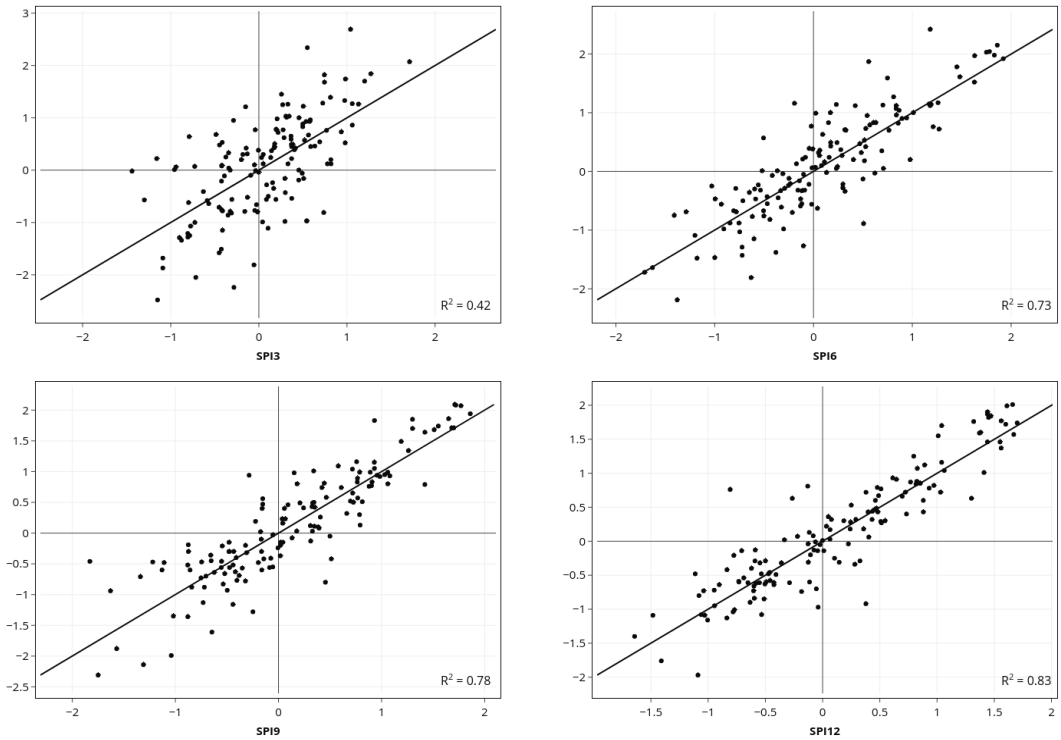

Figure 4.6: Scatter plot of SPI3, SPI6, SPI9, and SPI12

The criteria use to evaluate the error between target values and predict values is RMSE, the value of RMSE of SPI3, SPI6, SPI9, and SPI12 are 0.5236, 0.2225, 0.1812, and 0.1291, respectively. From the values of RMSE, we found that the long term SPI of 12 months has smallest error.

Table 4.14: The accuracy of each SPI

SPI	Correct	Incorrect	Accuracy
SPI3	101	41	71.13 %
SPI6	111	30	78.72%
SPI9	108	32	77.14%
SPI12	113	27	80.71%

Furthermore, scatter plots will be considered to evaluate the model which is shown in Figure 4.6. There are four scatter plots which x-axis represents the predict values and y-axis represents the target values. For SPI3, SPI6, SPI9, and

SPI12, the R-square are 0.42, 0.73, 0.78, and 0.83, respectively.

Table 4.14 shows the accuracy of each SPI. The results show that, the long term SPI of 12 months makes more accurate prediction than the short term SPI of 3, 6, and 9 months.

4.6 Conclusions

To study the scientific measurement of drought condition, in this research, we used standardized precipitation index (SPI) as a drought indicator to transform rainfall data from Nakhon Ratchasima province into drought index. The are several time-scales of standardized precipitation index. In this research, we focused on SPI3, SPI6, SPI9,and SPI12.

The different time-scale of standardized precipitation index was used to learn by a deep belief network with restricted Boltzmann machines. Since there is no mature method to determine architecture of a deep belief network, experimental method was used to find the suitable architecture to predict drought.

The suitable architecture of a deep belief network of each SPI which contained the smallest root mean square error will be used to predict drought. The result of predicting showed that the long term SPI of 12 months makes more accurate prediction than the short term SPI of 3, 6, and 9 months.

In view of the farmers in Nakhon Ratchasima province, the main occupation of the inhabitants is farming. The main majority of those farms produce rice. In the process of planting rice, rice needs a lot of water to survive between July to September. Therefore, the farmers should know the information that between

these months the water will be enough. To monitor drought in these three months, the short term SPI of 3 months is a good time-scales to predict drought. The best network structure of SPI3 is 3-25-15-1, it means that we need to know the values of monthly rainfall data from February to June in the same year of predicting drought. Then we can predict drought in July to September. Furthermore, SPI6, SPI9, SPI12 might be good to predict drought for the farmers who plant cassava, fruit orchard, and other tree which plants do not need a lot of water in specific month.

4.7 Future work

Since there is no mature method to determine architecture of a deep belief network, the best structure in this research was obtained from the experimentation. If there is a method to decide the suitable architecture of a deep belief network, the result of predicting drought might increase the accuracy.

References

Abramowitz, Milton, and Stegun. 1965. Handbook of mathematical functions: with formulas, graphs, and mathematical tables. 55. Courier Corporation.

Achutuni, Steyaert, and Sakamoto. 1982. Agroclimatic assessment methods for drought/flood shortages in South and Southeast Asis-Test and evaluation: Final Report to the Agency for International Development US of Foreing DisasterAssistance, washington.

Box, George, and Jenkins. 1976. Time series analysis: forecasting and control, revised ed. Holden-Day.

Byun, Hi-Ryong, and Wilhite. 1999. Objective quantification of drought severity and duration. Journal of Climate 9(12): 2747-2756.

Centre for Hydrology Irrigation Northeastern Lower. 2016. Rainfall Data. Available online: http://hydro-4.com/3rainfalldata/rainmonth/rainmonth.htm (October 26, 2016)

Chen, Hsin, and Murray. 2003. Continuous restricted Boltzmann machine with an implementable training algorithm. IEE Proceedings-Vision, Image and Signal Processing 3(150): 153-158.

Chen, Junfei, Jin, and Chao. 2012. Design of deep belief networks for short-term prediction of drought index using data in the Huaihe river basin.

Mathematical Problems in Engineering (2012): 1-16.

Crone and Nikolopoulos. 2007. Results of the NN3 neural network forecasting competition. In The 27th International Symposium on Forecasting, 129.

George. 1989. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems 4(2): 303-314.

Han, Ping, Wang, Zhang, and Zhu. 2010. Drought forecasting based on the remote sensing data using ARIMA models. Mathematical and Computer Modelling 11-12(51): 1398-1403.

Hecht. 1992. Theory of the backpropagation neural network. In Neural networks for perception :65-93.

Guttman. 1999. Accepting the standardized precipitation index: A calculation algorithm1. JAWRA Journal of the American Water Resources Association 2(35): 311-322.

George. 1975. Moisture availability and crop production. Transactions of the ASAE 5(18): 980-0984.

Hayes. 2006. Drought Indices. Available online: http://www.drought.unl.edu/whatis/indices.htm (October 26, 2016).

Hinton. 2002. Training products of experts by minimizing contrastive divergence. Neural computation 8(14): 1771-1800.

Hinton, Geoffrey, and Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. science 5786(313): 504-507.

Hopfield. 1984. Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the national

academy of sciences 10(81): 3088-3092.

Hrasko, André, and Krohling. 2015. Time series prediction using restricted Boltzmann machines and backpropagation. In ITQM, 990-999.

Hyndman, Rob, and George Athanasopoulos. 2018. Forecasting: principles and practice. OTexts.

Kumar. 2008. Time Series and its importance for business. Available online: http://mathematics.svtuition.org/2008/10/time-series-and-its-importance-for.html (October 26, 2016).

Lohani, Vinod, and Loganathan. 1997. An early warning system for drought management using the palmer drought index. Journal of the American Water Resources Association 6(33): 1375-1386.

McKee, Thomas, Nolan, and John. 1993. The relationship of drought frequency and duration to time scales. In Proceedings of the 8th Conference on Applied Climatology 22(17): 179-183.

Qi. 2007. Complete monotonicity of logarithmic mean. Research report collection 1(10).

SCB Economic Intelligence Center (SCB EIC). 2016. Thailand's Drought Crisis 2016: Understanding it without the Panic. Available online: https: //www.scbeic.com/en/detail/ product/2127 (October 28, 2016)

Shen, Furao, Jing, and Jinxi. Forecasting exchange rate using deep belief networks and conjugate gradient method. Neurocomputing (167): 243-253.

Smolensky. 1986. Information processing in dynamical systems: Foundations of harmony theory. No. CU-CS-321-86. Colorado UNIV at boulder

dept of computer science.

Svoboda and Wood. 2012. Standardized precipitation index user guide. World Meteorological Organization Geneva, Switzerland :1-16.

Thom. 1958. A note on the gamma distribution. Monthly weather review 4(86): 117-122.

Thorn. 1966. Some methods of climatological analysis. WMO technics/note number 81: 16-22.

Yaffee, Alan, and McGee. 2000. An introduction to time series analysis and forecasting: with applications of SAS® and SPSS®. Elsevier.

Appendix A

Restricted Boltzmann machine is an energy-based model which associates with the scalar value of the network, namely energy. Therefore, the energy function of restricted Boltzmann machine will be constructed. Configuration of a restricted Boltzmann machine is similar to a Hopfield network, except for that there is no connection between neurons in visible layer and hidden layer. Thus, the energy function of binary Hopfield network will be considered. To formulate the energy function of restricted Boltzmann machine using the energy of binary Hopfield network, we must modify the structure of Hopfield network to obtain the structure of restricted Boltzmann machine.

Suppose that we modify the structure of a Hopfield network (see Figure 3.5) to obtain the structure of restricted Boltzmann machine (see Figure 3.7) by exclusively dividing neurons in the network into two groups. The first group is visible layer which consists of m neurons, they are $N_1, N_2, N_3, ..., N_m$. The second group is hidden layer which consists of n neurons, they are $N_{m+1}, N_{m+2}, N_{m+3}, ..., N_p$, when $n = p - m$. To get the energy function of restricted Boltzmann machine, the

energy function of binary Hopfield network as Equation (3.17) will be considered.

$$
\begin{aligned}
E(u) &= -\frac{1}{2}\sum_{\substack{i=1\\ i\neq j}}^{p}\sum_{j=1}^{p} u_i t_{ij} u_j - \sum_{j=1}^{p} b_j u_j \\
&= -\frac{1}{2}\left(u_1 t_{12} u_2 + u_1 t_{13} u_3 + u_1 t_{14} u_4 + ... + u_1 t_{1m} u_m + ... + u_1 t_{1p} u_p\right) \\
&\quad - \frac{1}{2}\left(u_2 t_{21} u_1 + u_2 t_{23} u_3 + u_2 t_{24} u_4 + ... + u_2 t_{2m} u_m + ... + u_2 t_{2p} u_p\right) \\
&\quad - ... - \frac{1}{2}\left(u_m t_{m1} u_1 + u_m t_{m2} u_2 + u_m t_{m3} u_3 + ... + u_m t_{mp} u_p\right) \\
&\quad - ... - \frac{1}{2}\left(u_p t_{p1} u_1 + u_p t_{p2} u_2 + u_p t_{p3} u_3 + ... + u_p t_{pm} u_m + ... + u_p t_{pp-1} u_{p-1}\right) \\
&\quad - \left(b_1 u_1 + b_2 u_2 + b_3 u_3 + ... + b_m u_m + ... + b_p u_p\right)
\end{aligned}
$$

The weight t_{ij} is symmetric weight with $t_{ii} = 0$ and $t_{ij} = t_{ji}$. Therefore, we obtain

$$
\begin{aligned}
E(u) &= -\frac{1}{2}\left(2u_1 t_{12} u_2 + 2u_1 t_{13} u_3 + 2u_1 t_{14} u_4 + ... + 2u_1 t_{1m} u_m + ... + 2u_1 t_{1p} u_p\right) \\
&\quad - \frac{1}{2}\left(2u_2 t_{23} u_3 + 2u_2 t_{24} u_4 + ... + 2u_2 t_{2m} u_m + ... + 2u_2 t_{2p} u_p\right) \\
&\quad - ... - \frac{1}{2}\left(2u_m t_{mm+1} u_{m+1} + ... + 2u_m t_{mp} u_p\right) \\
&\quad - ... - \frac{1}{2}\left(2u_{p-1} t_{p-1p} u_p\right) \\
&\quad - \left(b_1 u_1 + b_2 u_2 + b_3 u_3 + ... + b_m u_m + ... + b_p u_p\right) \\
&= -\left(u_1 t_{12} u_2 + u_1 t_{13} u_3 + u_1 t_{14} u_4 + ... + u_1 t_{1m} u_m + ... + u_1 t_{1p} u_p\right) \\
&\quad - \left(u_2 t_{23} u_3 + u_2 t_{24} u_4 + ... + u_2 t_{2m} u_m + ... + u_2 t_{2p} u_p\right) \\
&\quad - ... - \left(u_m t_{m(m+1)} u_{m+1} + ... + u_m t_{mp} u_p\right) \\
&\quad - ... - \left(u_{p-1} t_{(p-1)p} u_p\right) \\
&\quad - \left(b_1 u_1 + b_2 u_2 + b_3 u_3 + ... + b_m u_m + ... + b_p u_p\right).
\end{aligned}
$$

Since restricted Boltzmann machine is not connected in the same layer (see Figure 3.7). Therefore, all connection between two visible neurons and all connection between two hidden neurons will be removed. In another word, the weight value $t_{ij} = 0$ when $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant m$ or $m+1 \leqslant i \leqslant p$ and $m+1 \leqslant j \leqslant p$. Then, we get

$$\begin{aligned} E(u) = & -\left(u_1 t_{1m+1} u_{m+1} + u_1 t_{1m+2} u_{m+2} + u_1 t_{1m+3} u_{m+3} + ... + u_1 t_{1p} u_p\right) \\ & -\left(u_2 t_{2m+1} u_{m+1} + u_2 t_{2m+2} u_{m+2} + u_2 t_{2m+3} u_{m+3} + ... + u_2 t_{2p} u_p\right) \\ & - ... - \left(u_m t_{m(m+1)} u_{m+1} + ... + u_m t_{mp} u_p\right) \\ & -\left(b_1 u_1 + b_2 u_2 + b_3 u_3 + ... + b_m u_m + ... + b_p u_p\right). \end{aligned}$$

To simplify this energy function ($E(u)$), we will define $v_i = u_i$, $a_i = b_i$, $h_j = u_{m+j}$, $b_j = b_{m+j}$, and $w_{ij} = t_{i,m+j}$. Furthermore, restricted Boltzmann machine consists of two layers. It is appropriate to use the energy of joint configurations of visible and hidden variables. Therefore, the energy function of restricted Boltzmann machine is

$$\begin{aligned} E(v,h) = & -\left(v_1 w_{11} h_1 + v_1 w_{12} h_2 + v_1 w_{13} h_3 + ... + v_1 w_{1n} h_n\right) \\ & -\left(v_2 w_{21} h_1 + v_2 w_{22} h_2 + v_2 w_{23} h_3 + ... + v_2 w_{2n} h_n\right) \\ & - ... - \left(v_m w_{m1} h_1 + ... + v_m w_{mn} h_n\right) \\ & -\left(a_1 v_1 + a_2 v_2 + a_3 v_3 + ... + a_m v_m\right) \\ & -\left(b_1 h_1 + b_2 h_2 + b_3 h_3 + ... + b_n h_n\right). \end{aligned}$$

We can write the simplified energy function of restricted Boltzmann machine in a scalar form as

$$E(v,h) = -\sum_{i=1}^{m}\sum_{j=1}^{n} v_i w_{ij} h_j - \sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j. \tag{4.4}$$

Appendix B

To derive the energy function of continuous restricted Boltzmann machine, we will consider the energy function of continuous Hopfield network as in Equation (3.18). Since we will derive the energy function of continuous restricted Boltzmann machine using the energy function of continuous Hopfield network, the structure of Hopfield network must be modified to obtain the structure of continuous restricted Boltzmann machine.

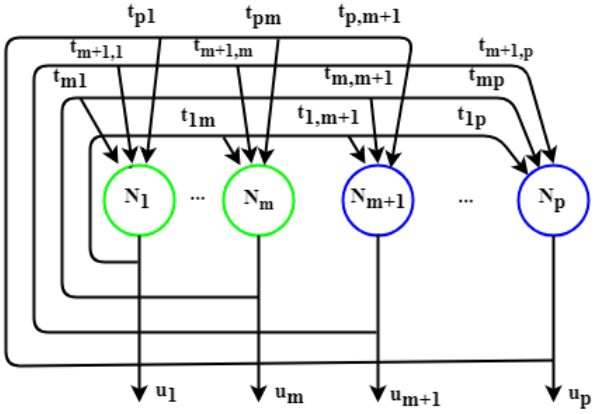

Figure 4.7: Hopfield network (separate neurons into two groups)

As the neuron in Hopfield network consists of p neurons, they are $N_1, N_2, N_3, ..., N_p$. These neurons will be divided into two groups. The first group consists of m neurons, they are $N_1, N_2, N_3, ..., N_m$. The second group consists of n neurons, they are $N_{m+1}, N_{m+2}, N_{m+3}, ..., N_p$, where $n = p - m$. Furthermore, we will define the bias value is zero, $b_i = 0$. Then we obtain the structure which is shown in Figure

4.7.

There is no connection in the same layer of continuous restricted Boltz-

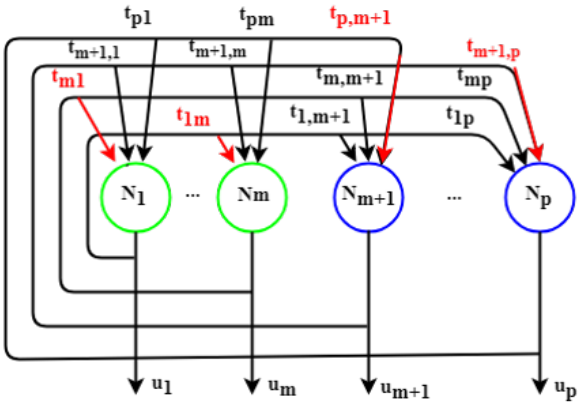

Figure 4.8: Hopfield network (remove some connection between two visible neurons and two hidden neurons)

mann machine. Therefore, all connections between two visible neurons and the connections between two hidden neurons will be removed. These connections are highlighted with red color in Figure 4.8

We will define the first group of neuron as $V_i = N_i, \leqslant i \leqslant m$. The second

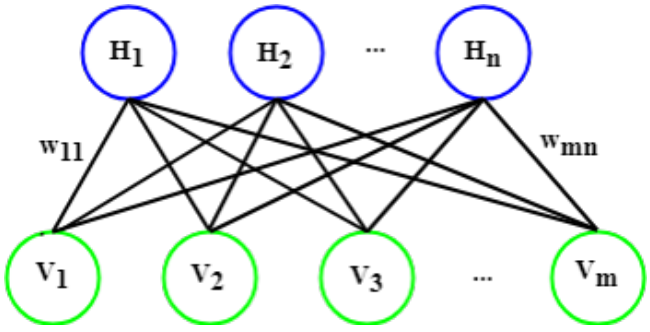

Figure 4.9: Continuous restricted Boltzmann machine (modified from Hopfield network)

group of neuron as $H_j = N_{m+j}, \leqslant j \leqslant n$. Inside of weight values, we will defined as $w_{ij} = t_{i,m+j}$. Furthermore, we will consider the transmembrane resistance R_j of neuron N_j as a bias value κ_i of visible neuron V_i and bias value λ_j of hidden neuron

H_j. Then we obtain the structure of continuous restricted Boltzmann machine as Figure 4.9. Therefore, the energy function of continuous restricted Boltzmann machine can be computed as Equation (4.5)

$$\begin{aligned} E(v,h) = &-\sum_{i=1}^{m}\sum_{j=1}^{n} v_i w_{ij} h_j + \sum_{i=1}^{m} \frac{1}{\kappa_i} \int_0^{v_i} \varphi^{-1}(v)dv \\ &+ \sum_{j=1}^{n} \frac{1}{\lambda_j} \int_0^{h_j} \psi^{-1}(h)dh, \end{aligned} \tag{4.5}$$

where

v_i is the output value of V_i,

w_{ij} is the weight value from V_i to H_j,

h_j is the output value of H_j,

κ_i is the value of noise control parameter of V_i,

$\varphi^{-1}(v)$ is the inverse of the activation function of V_i,

λ_j is the value of noise control parameter of H_j,

$\psi^{-1}(h)$ is the inverse of the activation function of H_j.

Appendix C

Gibbs sampling

Gibbs sampling is illustrated in Figure 4.10 which consists of k steps, each step consists of two phases. The first phase is positive phase and the second phase is negative phase, shown as Figure 4.11.

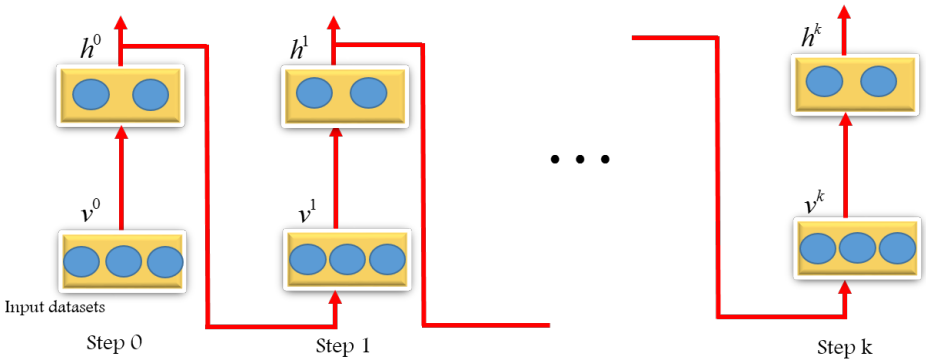

Figure 4.10: Gibbs sampling with k steps

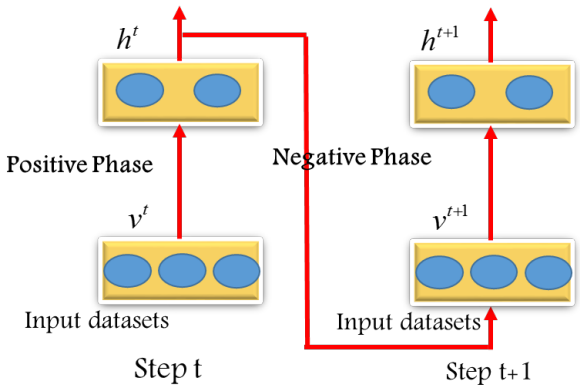

Figure 4.11: Each step of Gibbs sampling consists of two phase, positive phase and negative phase

Positive phase

Positive phase is initialized using the output value v_j in visible layer to calculate the input value s_j and output value h_j of the neuron in hidden layer at time t

$$s_j = \sum_{i=1}^{m} w_{ij} v_i + \sigma N_j(0,1),$$

with

$$h_j = \theta_L + (\theta_H - \theta_L) \frac{1}{1 + e^{-\lambda_j s_j}}.$$

Negative phase

Negative phase is used to calculate the input value r_j and the output value v_j of the neural in visible layer at time $t+1$. This is possible since the value of neural in hidden layer at time t is known as

$$r_j = \sum_{i=1}^{n} w_{ij} h_i + \sigma N_j(0,1)$$

with

$$v_j = \theta_L + (\theta_H - \theta_L) \frac{1}{1 + e^{-\kappa_j r_j}}. \tag{4.6}$$

After calculating the value of visible neuron V_j and hidden neuron H_j at step 0, $\left\langle v_i^{(0)} h_j^{(0)} \right\rangle$ denoted by $wpos_{ij}$ and $\left\langle \left(h_j^{(0)}\right)^2 \right\rangle$ denoted by $apos_j$ will be

calculated using Equations (4.7) and (4.8), respectively

$$wpos_{ij} = v_i^{(0)} h_j^{(0)}, \tag{4.7}$$

$$apos_{ij} = (h_j^{(o)})^2. \tag{4.8}$$

This process will be repeated by alternating positive and negative phases until step k. At step k, $\left\langle v_i^{(k)} h_j^{(k)} \right\rangle$ denoted by $wneg_{ij}$ and $\left\langle (h_j^{(k)})^2 \right\rangle$ denoted by $aneg_j$. These two values can be computed by Equations (4.9) and (4.10), respectively

$$wneg_{ij} = v_i^{(k)} h_j^{(k)}, \tag{4.9}$$

$$aneg_{ij} = (h_j^{(k)})^2. \tag{4.10}$$

From the previous process, $wpos_{ij}$, $wneg_{ij}$, $apos_j$, and $aneg_j$ were calculated. By these calculations, weights and noise control parameter will be updated. These values of two parameters will keep updating using the next set of data points until finish all data. The optimal weight values in this path will be initial weight values in supervised path.

Vitae

Name :Sureeluk Ma

Student ID :5820320706

Educational Attainment:

Degree	**Name of Institution**	**Year**
B.Sc. (Applied Mathematics)	Prince of Songkha University	2014

Scholarship Award during Enrolment

Centre of Excellence in Mathematics, Commission on Higher Education (CHE), Ministry of Education, Ratchathewi, Bangkok, Thailand.

Publications and Proceedings

Sureeluk, Pakwan, and Tatdow. 2018. Mathematical Derivations of the Energy Function and Parameter's Update Rules for a Continuous Restricted Boltzmann Machine. The 23rd Annual Meeting in Mathematics. Mandarin Hotel, Bangkok, May 3-4, 2018, 272-277.