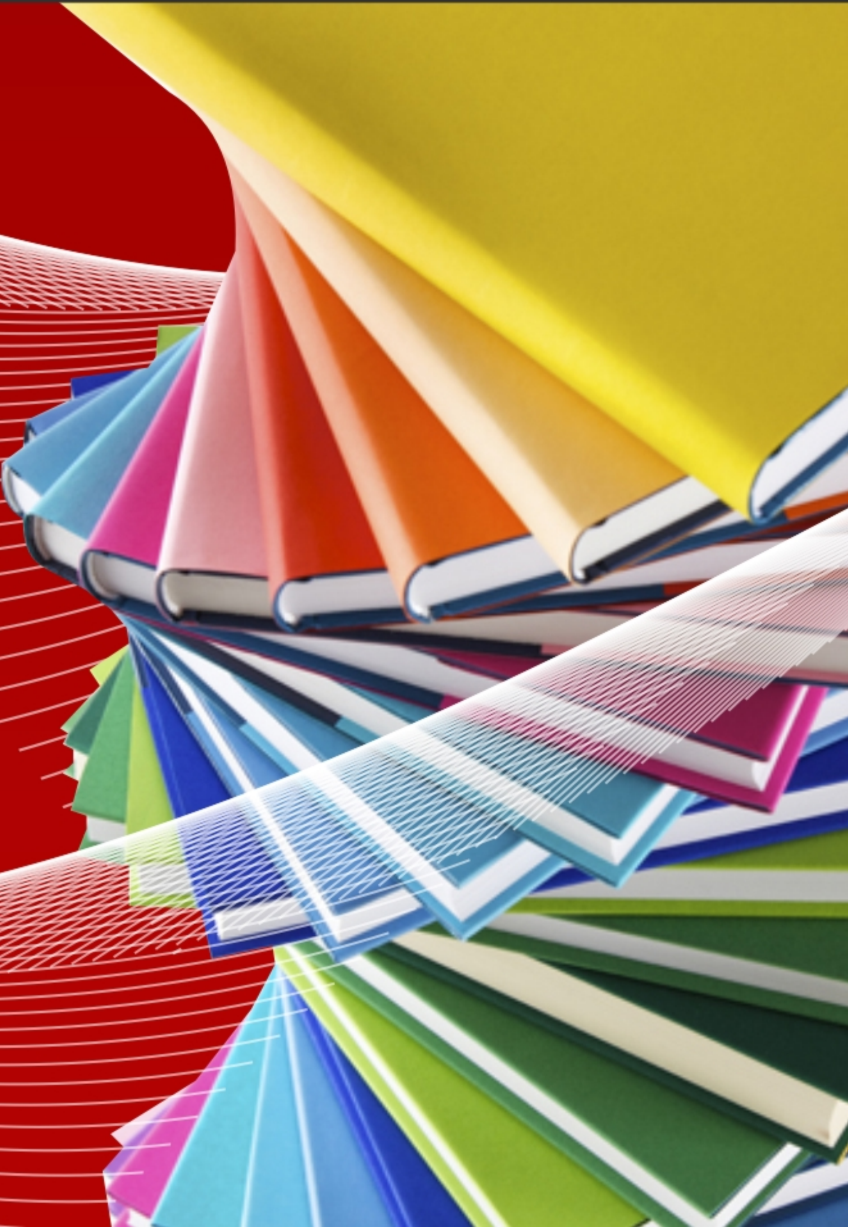
澳門大學
UNIVERSIDADE DE MACAU
UNIVERSITY OF MACAU

Outstanding Academic Papers by Students
學生優秀作品



University of Macau

Faculty of Science and Technology

澳 門 大 學

UNIVERSIDADE DE MACAU

UNIVERSITY OF MACAU

A Distributed Implementation of Training the Restricted Boltzmann Machine

by

Kin Tek NG, Student No: D-B0-2843-2

Final Project Report submitted in partial fulfillment
of the requirements of the Degree of
Bachelor of Science in Software Engineering

Project Supervisor

Prof. C. L. Philip CHEN

08 October 2014

DECLARATION

I sincerely declare that:

1. I and my teammates are the sole authors of this report,
2. All the information contained in this report is certain and correct to the best of my knowledge,
3. I declare that the thesis here submitted is original except for the source materials explicitly acknowledged and that this thesis or parts of this thesis have not been previously submitted for the same degree or for a different degree, and
4. I also acknowledge that I am aware of the Rules on Handling Student Academic Dishonesty and the Regulations of the Student Discipline of the University of Macau.

Signature : ______________________________

Name : Kin Tek NG

Student No. : D-B0-2843-2

Date : 08 October 2014

ACKNOWLEDGEMENTS

The author would like to express his utmost gratitude to UM for providing the opportunity to carry out a project as a partial fulfillment of the requirement for the degree of Bachelor of Science.

Throughout this project, the author was very fortunate to receive the guidance and encouragement from his supervisor…

ABSTRACT

In these recent years, deep learning technique becomes very important in the artificial intelligence research, especially in the machine learning field. Deep learning works well in different applications in machine learning such as image, speech, document processing, etc. Since deep learning is related to a lot of mathematical calculations. Some well-known mathematical model running in behind of it, so it is hard to get start as a novice. As most of deep architectures [1], such as, Deep Belief Network (DBN) [2], Deep Boltzmann Machine [3], stacked auto-encoder [4], are related to or based on the Restricted Boltzmann Machine (RBM) [5]. In this report, we are focus on the training process [6] and distributed implementation of training the Restricted Boltzmann Machine, also evaluating the performances of Restricted Boltzmann Machine in distributed environment.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1.1 Background

Restricted Boltzmann machine (RBM) is a two-layer model (see Figure 1), which consists of a lot of nodes, we call them Neurons. Each node has a connection with every node in the other layer. Each neuron has its own biases, we usually use the *v* for represent the input data, and/or we call it visible nodes. In the other layer, we use *h* for represent the features, and/or we call it hidden nodes. Also, there is a weight in each connection, we use of *w* to represent it.

RBM is a generative stochastic graph model, which can be used to learn the probability distribution [7] of unlabelled data from scratch. Therefore, it has another property, which is to regenerate some samples from the probability distribution.

And how does it work? In fact it is really simple, as we said, this model can be used to learn the probability, and generate some samples from the probability distribution. This means that after our model has been well trained, the reconstruction samples which were regenerated from the distribution should be almost the same as the original data.

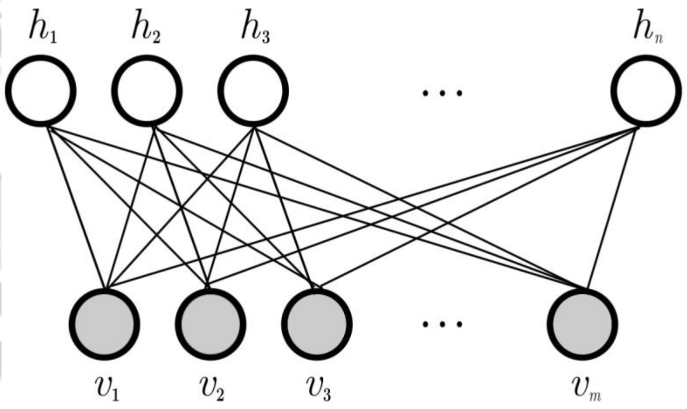


Figure 1: The structure of Restricted Boltzmann Machine

1.2 Objective

We can find out the different in between the original data and the reconstruction data, and our work is try to keep this value very small, which does somehow like to find the minimum of a curve. We can use the gradient descend method easily to find out the local minimum of the curve, so does the RBM. But there is another problem comes out immediately. If we only have one curve, the minimum value can be found easily by make use of the gradient descend method. How about if we have many different curves, and want to find out a common minimum value. In the RBM, we are almost at the same situation.

How to find out a parameter set can satisfy is a big challenge for us. Indeed, there are thousands or even millions of parameters inside the deep model, how to use the gradient descend method to solve this kind of question will be discuss in this report.

CHAPTER 2. PRELIMINARY

In this report, there are a lot of related mathematical, probability concepts. We will discuss of them one by one here.

2.1 Mathematics

2.1.1Exponential Function

Exponential is a mathematical operation, written as b^n, involving two numbers, the base *b* and the exponent (or power) *n*. When the base b will equal to Euler's number, we donate it as e, is an important mathematical constant and can be calculated as

$$\mathrm{e} = \lim_{\mathrm{n}\to\infty}\left(1+\frac{1}{\mathrm{n}}\right)^{\mathrm{n}} = \sum_{\mathrm{n}=0}^{\infty}\frac{1}{\mathrm{n}!}. \tag{2.1}$$

When the base is the Euler's number, and the exponent is n, we written it as *exp(n)*.

There is some important property for the exponential such as the associativity of multiplication is shown as

$$b^{m=n} = b^m b^n. \tag{2.2}$$

2.1.2Logarithm Function

The logarithm function is a function related to the exponential function. The logarithm function and exponential function has some relation in between is shown as

$$x = b^n \iff n = \log_b x, \tag{2.3}$$

$$\mathrm{n} = \log_{\mathrm{b}} \mathrm{b}^{\mathrm{n}} = \mathrm{b}^{\log_{\mathrm{b}} \mathrm{n}}. \tag{2.4}$$

Where *b* is also call the base. When the base is equal to 2, we call it binary logarithm. When the base is equal to 10, we call it common logarithm, we written it as *lg(n).* When the base is equal to Euler's number, we call it nature logarithm, we written it as *ln(n).*

There are some important properties for the logarithm such as the associativity of multiplication and division is shown as

$$\log_b xy = \log_b x + \log_b y, \tag{2.5}$$

$$\log_b \frac{x}{y} = \log_b x - \log_b y. \tag{2.6}$$

2.1.3Derivative

The derivative of a function of a real variable measures the sensitivity to change of a quantity *f(x)* or *y* (a function or dependent variable) which is determined by another quantity *x* (the independent variable). We denote it as *df(x)/dx* or *dy/dx*. For example,

$$\frac{\mathrm{d}}{\mathrm{dx}}e^{x}=e^{x}\frac{d}{dx}(x)=e^{x}, \tag{2.7}$$

$$\frac{\mathrm{d}}{\mathrm{dx}}ln(x)=\frac{1}{x}\frac{d}{dx}(x)=\frac{1}{x}. \tag{2.8}$$

2.1.4Partial Derivatives

Assumed that our function is related to more than one variable, we can do the derivatives in each of the variable. The other variable we see as a constant value during the partial derivatives. Assume that we have a function f is relate to two variable x and y, we can denote the partial derivatives of f as $\partial f(x,y)/\partial \mathrm{x}$ and $\partial f(x,y)/\partial \mathrm{y}$.

2.1.5Stochastic Gradient Descent

Usually in both the statistical estimation and machine learning will face a problem about minimizing an objective function, which is in a form of summarization. Like

$$Min_{\theta}J=\sum_{i=1}^{n}f(\theta,x_i). \tag{2.9}$$

Where J is the problem we want to find out the minimize value, x_i is the samples we use, θ is the parameter inside the function f, n is the number of samples.

In order to solve this problem, we can use the stochastic gradient descent [8][9] method to solve it. The update rule of the parameter θ is

$$\theta=\theta-\alpha\nabla J=\theta-\alpha\nabla\sum_{i=1}^{n}f(\theta,x_i). \tag{2.10}$$

2.2 Probability

2.2.1Joint Distribution

When we have at least two random variables define in one probability distribution, and each of the variable have their own given probability distribution. Joint distribution is a probability distribution combination of that each of variable falls in any particular range or discrete set of values specified for that variable.

In general, the joint probability distribution of n discrete random variables x_1, x_2 … x_n is equal to

$$\mathrm{P}(x_1,x_2,\dots,x_n)=\mathrm{P}(x_1)\times\mathrm{P}(x_2|x_1)\times\dots\times\mathrm{P}(x_n|x_1,x_2,\dots,x_{n-1}). \tag{2.11}$$

2.2.2Marginal Distribution

The marginal distribution of subset of a collection of random variables is the probability distribution of the variables contained in the subset. The marginal probability can be written as a summation of joint probability, for example, probability P is define in two different variable spaces X and Y as

$$\mathrm{P}(X = x) = \sum_{y} P(X = x, Y = y) = \sum_{y} P(X = x | Y = y) \times P(Y = y). \qquad (2.12)$$

2.2.3Conditional Distribution

The conditional probability distribution when we have at least two random variable spaces defines in the probability distribution. We want to find out the probability distribute under some given variables, for example, probability P is define in two different variable spaces X and Y. We want to find out the probability X given by Y as

$$\mathrm{P(X = x|Y = y)} = \frac{P(X = x, Y = y)}{P(X = x)}. \qquad (2.13)$$

2.3 Graphical Model

A graphical model is a probabilistic model for which a graph denotes the conditional dependence structure between random variables. Mainly have two branches: directed acyclic graphical model and undirected graphical model, and the representation of them are Bayesian networks and Markov networks.

2.3.1Markov Random Field

Markov random field [10] is a set of random variables having a Markov property described by an undirected graph and may be cyclic. Thus, a Markov network can represent certain dependencies**.**

2.3.2Generative Model

A generative model [11] is a model for randomly generating from some observable data, typically contain some hidden parameters. It specifies a joint probability distribution over the observable data and the hidden parameters.

2.4 Machine Learning

2.4.1Supervised Learning

We can use the supervised learning to label some training data. Usually using the label combines with the data to build an inferred function, this can be predicting the label of the new examples. One of the examples is the classification problem, all the data will separate in different categories. The commonest one is separate the data in two categories, mainly use -1/+1 to represent which group does the data belong to.

2.4.2Unsupervised Learning

The unsupervised learning is trying to find out the hidden structure of the data, such as the relation in between different dimensions of the data. It is very hard to evaluate the model except we have the label of the data before the training. One of the examples is the clustering problem, which is trying to separate the data into different groups systematically. In the same group will have some common factors, these common factors are learned from the hidden structure.

CHAPTER 3. ENERGY-BASED MODELS

Energy-based model (EBM) is a probabilistic model, which associate a scalar energy to each configuration of the variables of interest. A configuration is combined with a lot of variables, for example, $x=\{x_1, x_2, x_3 \ldots x_m\}$, there are m variable inside the configuration x. If each variable can be assigned two possible values, the number of possible configuration will be 2^m.

3.1 Model Establishment

3.1.1Energy Function

As we said before, each configuration is composed by a lot of variable, and which is difficult to measurement. So that we have to design a function for all the configurations, that can convert each of the configuration into a value. We call this function as an energy function, and denoted it as *Energy(x)* for the configuration x. In addition, this energy function is bounded by a large set of parameters. We denote this set of parameters as θ.

3.1.2Probability Distribution

Assume that we have N different configurations x^1, x^2 ... x^N in domain D, each of the configuration can be convent to a value by make use of the energy function. Therefore, we can calculate the energy value of all the configurations in domain D.

In addition, the value that converted by energy function may not be positive, then we may cancel out part of the energy value during calculate the summation in between energy values. What we do to settle this problem is to put the value into the exponential function.

In fact, most of our energy will provide a negative value, what we want is the smallest energy value will have the largest probability. Therefore, we will first apply the negative sign in the value before we put it into the exponential function. Let us have summarize by make use of (see Figure 2) to explain it, when the Energy function return a small value, it will become very large, which magnify the negative value of the energy.

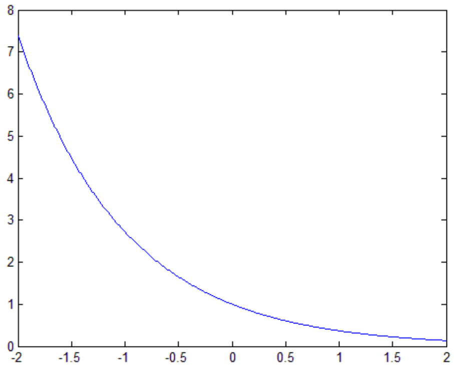


Figure 2: Equation of f(x) = exp(-x)

We define the probability distribution of each configuration as that the energy value of the particular configuration over the summation energy value of all configurations. The probability distribution is denoted as

$$p(x) = \frac{exp(-Energy(x))}{\sum_{x^i \in D} exp(-Energy(x^i))}. \tag{3.1}$$

3.1.3Partition Function

We call the summation over all configurations as a normalizing factor, or the partition function *Z*. We can use that to simplify the formula as

$$Z = \sum_{x^i \in D} exp(-Energy(x^i)), \tag{3.2}$$

$$P(x) = \frac{exp(-Energy(x))}{Z}. \tag{3.3}$$

3.2 Energy-Based Models with Hidden Variable

3.2.1Introduction

In many cases, the observed variables cannot deeply represent the sample. Therefore, we have to introduce some new variables to increase the representation power of the model. The model can be improved by adding some hidden variables into the original model.

The hidden variables is not part of the observed variables, it is not visible for us, which we call it hidden nodes. But it plays an important role in calculate the energy value. At the same time, the hidden variables are depended on the observed variables. The hidden variables will be changing in different time under a particular configuration.

We denoted the observed part of the configuration as *x*, and the non-observed part or hidden variables as *y*. The energy function will accept *x* and *y* as its parameter, we rewrite the energy function as *Energy(x, y)*.

3.2.2Probability Distribution

Now we apply the change of energy function into the distribution of the model. The probability has a different form Eq. (3.3) as

$$P(x, y) = \frac{exp(-Energy(x, y))}{Z}. \tag{3.4}$$

Where p(x=x^i, y=y^j) is the joint distribution of the variables when x = x^i and y = y^j. But in fact, we do not care it so much. What we want is only the probability distribution of the observed variables, since we cannot control the non-observed variables. Therefore, we have better to convert the probability of the variable into a form that only contains the observed variables. The easiest way is to calculate the marginal distribution as

$$P(x) = \sum_{\tilde{y}} P(x, \tilde{y}) = \sum_{\tilde{y}} \frac{exp(-Energy(x, \tilde{y}))}{Z}. \tag{3.5}$$

3.2.3Free Energy

We can see the Eq. (3.5) is very complicated and difficult to reuse for the future calculation, so that we have better to change the representation. We will introduction (inspired from physics) of free energy $\mathcal{F}\,(x)$, which we wanted to change the probability distribution format as the same as before. Then, we have

$$P(x) = \frac{exp(-\mathcal{F}(x))}{Z}; \tag{3.6}$$

$$\mathcal{F}(x) = \ -ln \sum_{\tilde{y}} exp\big(-Energy(x, \tilde{y})\big). \tag{3.7}$$

And now we can try to find out how to get the Eq. (3.7), we can derive the equation of free energy $\mathcal{F}(x)$ from Eq. (3.5-3.6) as

$$\frac{exp(-\mathcal{F}(x))}{Z} = \sum_{\tilde{y}} \frac{exp(-Energy(x, \tilde{y}))}{Z}$$

$$exp(-\mathcal{F}(x)) = exp\Big(ln \textstyle\sum_{\tilde{y}} exp\big(-Energy(x, \tilde{y})\big)\Big)$$

$$\mathcal{F}(x) = \ -ln \sum_{\tilde{y}} exp\big(-Energy(x, \tilde{y})\big).$$

At the same time, the partition function becomes

$$Z = \sum_{\tilde{x}} exp(-\mathcal{F}(\tilde{x})). \tag{3.8}$$

CHAPTER 4. MODEL OPTIMIZATION

4.1 Objective Function

The object of the energy-based models is try to find out the parameter θ, which can makes every sample of configuration $x^1, x^2 \ldots x^N$ can get a large amount of probability.

Since all the samples of configuration do not have any relation, we can easy to do the multiplication in between them. Try to find out the parameter set can provide the maximum values of all the possible samples, which does also the objective function we want to find out. The objective function $\mathcal{L}$ can be define as

$$\mathcal{L} = P(x^1)P(x^2)\ldots P(x^N)$$

$$= \prod_{x^i \in D} P\left(x^i\right). \tag{4.1}$$

But the multiplication inside the Eq. (4.1) is very complicate to calculate, we have to change a little bit for this objective function, such as apply it into the natural logarithm function. Since the natural logarithm function is a monotonically increasing function, it would not affect the convergence of the parameter θ, and our problem will become

$$Max_\theta \mathcal{L} = Max_\theta\, ln\left(P(x^1)P(x^2)\ldots P(x^N)\right)$$

$$= Max_\theta ln\left(P(\mathrm{x}^1)\right) + ln\left(P(\mathrm{x}^2)\right) + \cdots + ln\left(P(\mathrm{x}^N)\right)$$

$$= Max_\theta \sum_{x^i \in D} ln\left(P\left(x^i\right)\right). \tag{4.2}$$

Now the equation becomes simpler, but it is hard to calculate a maximization problem at all. In order to solve it, we have better to convent the problem to a minimization problem. The conversion is shown as

$$Min_\theta \ell = -Max_\theta \mathcal{L} = Min_\theta \sum_{x^i \in D} -ln\left(P\left(x^i\right)\right). \tag{4.3}$$

4.2 Stochastic Gradient Descent

We notice that the Eq. (4.3) can be solved by make use of the stochastic gradient descent optimization method. The update rule of the parameter is

$$\theta = \theta - \alpha \nabla \ell(\theta, D)$$

$$= \theta - \alpha \sum_{x^i \in D} -\frac{\partial ln\left(P\left(x^i\right)\right)}{\partial \theta}. \tag{4.4}$$

Where α is the step size or we called it as the learning rate in machine learning.

We can see there is a key value inside the stochastic gradient descent process, but it seems very difficult to calculate. We take it out to do more calculation and translation, until it can get a meaningful form.

First step is to apply Eq. (3.6), since what we focus is in EBM with hidden variable. Make use of the property of logarithm function to do the conversion

$$\begin{aligned}
-\frac{\partial ln\left(P\left(x^i\right)\right)}{\partial\theta} &= -\frac{\partial ln \frac{exp\left(-\mathcal{F}\left(x^i\right)\right)}{Z}}{\partial\theta} \\
&= -\frac{\partial ln\, exp\left(-\mathcal{F}\left(x^i\right)\right)}{\partial\theta} + \frac{\partial\, ln\, Z}{\partial\theta} \\
&= \frac{\partial\mathcal{F}\left(x^i\right)}{\partial\theta} + \frac{\partial\, ln \sum_{\tilde{x}} exp(-\mathcal{F}(\tilde{x}))}{\partial\theta}. \qquad (4.5)
\end{aligned}$$

Secondly, calculate the partial derivative in the second part of the Eq. (4.5). We have to compute the derivative in the logarithm function, and then the function inside the logarithm function. The process is shown as

$$\begin{aligned}
\frac{\partial\, ln \sum_{\tilde{x}} exp(-\mathcal{F}(\tilde{x}))}{\partial\theta} &= \frac{\frac{\partial \sum_{\tilde{x}} exp(-\mathcal{F}(\tilde{x}))}{\partial\theta}}{\sum_{\hat{x}} exp(-\mathcal{F}(\hat{x}))} \\
&= \frac{\sum_{\tilde{x}} \frac{\partial exp(-\mathcal{F}(\tilde{x}))}{\partial\theta}}{\sum_{\hat{x}} exp(-\mathcal{F}(\hat{x}))} \\
&= -\frac{\sum_{\tilde{x}} exp\left(-\mathcal{F}(\tilde{x})\right) \frac{\partial\mathcal{F}(\tilde{x})}{\partial\theta}}{\sum_{\hat{x}} exp\left(-\mathcal{F}(\hat{x})\right)}. \qquad (4.6)
\end{aligned}$$

Thirdly, in the Eq. (4.6) we can see, the denominator $\sum_{\hat{x}} \exp\left(-\mathcal{F}(\hat{x})\right)$, which was the fixed value we defined in partition function Z. We can put it back into the summarization in the divisor, and also take out the derivative part. We can find out that there exists a pattern we see before in Eq. (3.6), so that we can use it to change the representation as

$$\begin{aligned}
\frac{\sum_{\tilde{x}} exp\left(-\mathcal{F}(\tilde{x})\right) \frac{\partial\mathcal{F}(\tilde{x})}{\partial\theta}}{\sum_{\hat{x}} exp\left(-\mathcal{F}(\hat{x})\right)} &= \sum_{\tilde{x}} \frac{exp\left(-\mathcal{F}(\tilde{x})\right)}{Z} \frac{\partial\mathcal{F}(\tilde{x})}{\partial\theta} \\
&= \sum_{\tilde{x}} P(\tilde{x}) \frac{\partial\mathcal{F}(\tilde{x})}{\partial\theta}. \qquad (4.7)
\end{aligned}$$

Finally, we combine the Eq. (4.5-4.7), and get the reasonable and meaningful formula:

$$-\frac{\partial ln\left(P\left(x^i\right)\right)}{\partial\theta} = \frac{\partial\mathcal{F}\left(x^i\right)}{\partial\theta} - \sum_{\tilde{x}} P(\tilde{x})\ \frac{\partial\mathcal{F}(\tilde{x})}{\partial\theta}. \tag{4.8}$$

From the Eq. (4.8), we can see there exists two terms. One is positive and one is negative. We can see the variable x has some different in between two terms, the positive term is in the format of observed variables that we define before. The negative term is in the format we define for the iterator, which may be anyone of the N possible observed variables in the domain D.

As we see carefully, we can find out that the $P(\tilde{x})$ is a probability distribution of $\tilde{x}$, and $\partial\mathcal{F}/\partial\theta$ is a function relate to $\tilde{x}$, so that we can find out the relation

$$E\left[\frac{\partial\mathcal{F}(x)}{\partial\theta}\right] = \sum_{\tilde{x}} P(\tilde{x})\ \frac{\partial\mathcal{F}(\tilde{x})}{\partial\theta}. \tag{4.9}$$

Weighted average can be used to calculate the expected value. But it is too hard to calculate the weighted average of all the samples. However, we know that the means value will almost surely converge to it.

The Means value can be approximate by a number of samples, which can help us easily to estimate the value of the expected value. Therefore, we can approximate convent Eq. (4.9) into the mean value format as

$$-\frac{\partial ln\left(P\left(x^i\right)\right)}{\partial\theta} \approx \frac{\partial\mathcal{F}\left(x^i\right)}{\partial\theta} - \frac{1}{|N|}\sum_{\tilde{x}\in D} \frac{\partial\mathcal{F}(\tilde{x})}{\partial\theta}. \tag{4.10}$$

As there are too many of possible variable $\tilde{x}$, how to choose the good samples is a big problem for us to handle.

4.3 Energy Function for Two-Layer Model

We assume that the energy function can be written as a sum of terms associated with at most one non-observed variables, which means that each of the non-observed variable do not have a relation with other non-observed variable. Also, the same restrict on the observed variables. Then, the energy function can be define as

$$Energy(x,y) = -\beta(x) + \sum_i \gamma_i(x,y_i). \tag{4.11}$$

We can see the energy function include two parts, the first part is only involve the observed variables, the second include the both the observed and non-observed variables. It is because of the observed variables can directly affect the energy value. The non-observed variables have to through the observed variables to affect the energy value.

Every hidden variable is associated with a corresponding γ function, and share the same observed variables. We can see all the observed variables will affect each non-observed variable independently.

Now we apply the new Eq. (4.11) into the model we define before, and put it into the probability distribution function Eq. (3.5). Since the conversion is so complicate, we take out the partition function and separate the summarization function into different parts. The distribution function is given by

$$
\begin{aligned}
P(x) &= \sum_{\tilde{y}} \frac{exp(-Energy(x,\tilde{y}))}{Z} \\
&= \frac{1}{Z} \sum_{y_1} \sum_{y_2} \dots \sum_{y_n} exp(-Energy(x,y)) \\
&= \frac{1}{Z} \sum_{y_1} \sum_{y_2} \dots \sum_{y_n} exp(\beta(x) - \sum_{i} \gamma_i(x,y_i)). \qquad (4.12)
\end{aligned}
$$

In Eq. (4.12) we can see that the function β do not include in the scope of y, so that we can take it out of the summarization scope. Secondly, the summarization of the γ function is inside of the exponential function. According to the associativity of multiplication of the exponential function, the addition in the exponent can be converted to the multiplication form as

$$
\begin{aligned}
&\sum_{y_1} \sum_{y_2} \dots \sum_{y_n} exp(\beta(x) - \sum_{i} \gamma_i(x,y_i)) \\
&= \sum_{y_1} \sum_{y_2} \dots \sum_{y_n} exp\big(\beta(x)\big)\, exp\big(-\gamma_1(x,y_1)\big) exp\big(-\gamma_2(x,y_2)\big) \dots exp\big(-\gamma_n(x,y_n)\big) \\
&= \sum_{y_1} \sum_{y_2} \dots \sum_{y_n} exp\big(\beta(x)\big) \left(\prod_{i} exp\big(-\gamma_i(x,y_i)\big) \right) \\
&= \; exp\big(\beta(x)\big) \sum_{y_1} \sum_{y_2} \dots \sum_{y_n} \prod_{i} exp\big(-\gamma_i(x,y_i)\big). \qquad (4.13)
\end{aligned}
$$

Next is to associate the items. We can see the scope of the summarization in Eq. (4.13) is only related to the item y_i. We can separate each item into the corresponding multiplier. Then, we can find the common factor and merge them together again. The process is shown as

$$
\begin{aligned}
&\sum_{y_1} \sum_{y_2} \dots \sum_{y_n} \prod_{i} exp\big(-\gamma_i(x,y_i)\big) \\
&= \sum_{y_1} exp\big(-\gamma_1(x,y_1)\big) \sum_{y_2} exp\big(-\gamma_2(x,y_2)\big) \dots \sum_{y_n} exp\big(-\gamma_n(x,y_n)\big) \\
&= \prod_{i} \sum_{y_i} exp\big(-\gamma_i(x,y_i)\big). \qquad (4.14)
\end{aligned}
$$

We combine the Eq. (4.12 - 4.14), and get a formula by make use of the energy function shown in Eq. (4.11) as

$$P(x) = \frac{exp\big(\beta(x)\big)}{Z} \prod_{i} \sum_{y_i} exp\big(-\gamma_i(x, y_i)\big). \tag{4.15}$$

Apply the Eq. (4.15) into the left hand side of the Eq. (3.6) and obviously can get the relation as

$$\frac{exp(-\mathcal{F}(x))}{Z} = \frac{exp\big(\beta(x)\big)}{Z} \prod_{i} \sum_{y_i} exp\big(-\gamma_i(x, y_i)\big)$$

$$ex\,p\big(-\mathcal{F}(x)\big) = exp\big(\beta(x)\big) \prod_{i} \sum_{y_i} exp\big(-\gamma_i(x, y_i)\big)$$

$$\mathcal{F}(x) = -ln\left(exp\big(\beta(x)\big) \prod_{i} \sum_{y_i} exp\big(-\gamma_i(x, y_i)\big) \right). \tag{4.16}$$

According to the associativity of multiplication of the logarithm function, we can have a further simplification in Eq. (4.16). In order to do that, we have to first extend the whole formula as

$$ln\left(exp\big(\beta(x)\big) \prod_{i} \sum_{y_i} exp\big(-\gamma_i(x, y_i)\big) \right)$$

$$= ln\left(exp\big(\beta(x)\big) \sum_{y_1} exp\big(-\gamma_1(x, y_1)\big) \sum_{y_2} exp\big(-\gamma_2(x, y_2)\big) \dots \sum_{y_n} exp\big(-\gamma_n(x, y_n)\big) \right)$$

$$= ln\Big(exp\big(\beta(x)\big)\Big) + ln\left(\sum_{y_1} exp\big(-\gamma_1(x, y_1)\big) \right) + ln\left(\sum_{y_2} exp\big(-\gamma_2(x, y_2)\big) \right) \dots$$

$$+ ln\left(\sum_{y_n} exp\big(-\gamma_n(x, y_n)\big) \right)$$

$$= \beta(x) + \sum_{i} ln\left(\sum_{y_i} exp\big(-\gamma_i(x, y_i)\big) \right). \tag{4.17}$$

We combine the Eq. (4.16 - 4.17) together, and get the formula $\mathcal{F}$ as

$$\mathcal{F}(x) = -\beta(x) - \sum_{i} ln\left(\sum_{y_i} exp\big(-\gamma_i(x, y_i)\big) \right). \tag{4.18}$$

CHAPTER 5. RESTRICTED BOLTZMANN MACHINE

5.1 EBM and RBM

In our previous work, we have defined the EBM for a special two layer model. Also, the RBM is a model with the same structure with it. What we want to do is to apply the EBM into the RBM.

Typically, we can use a quadratic function to represent a general two-layer model as

$$f(x,y) = Ax^2 + By^2 + Cxy + Dx + Ey + F. \tag{5.1}$$

Based on the structure of RBM, we can cancel out some useless coefficient and rename some the coefficient in Eq. (5.1). Moreover, we have to change data format into matrix for future calculation, change the function name into the energy function. Then the new representation become

$$Energy(v,h) = -b'v - c'h - h'Wv. \tag{5.2}$$

In the Eq. (5.2), we can see that b and c is the biases of the visible and hidden variables, both of them are column vector. W is the weights in between the visible and hidden variables.

In order to more suitable the two-layer energy-based model, we have to change the equation in the format of Eq. (4.11). The representation become

$$\begin{aligned} Energy(v,h) &= -(b'v) + h'(-c - Wv) \\ &= -(b'v) + \sum_{i=1}^{n} h_i(-c_i - W_i v). \end{aligned} \tag{5.3}$$

We compare the Eq. (4.11) and Eq. (5.3), found out that the function $\beta(x) = b'x$ and $\gamma_i(x, y_i) = y_i(-c_i - W_i x)$. We change the x into v, y into h. Then substitute them into the Eq. (4.18), get the new formula of free energy $\mathcal{F}$ as

$$\begin{aligned} \mathcal{F}(v) &= -\beta(v) - \sum_{i=1}^{n} ln\left(\sum_{h_i} exp\big(-\gamma_i(v,h_i)\big)\right) \\ &= -b'v - \sum_{i=1}^{n} ln\left(\sum_{h_i} exp\big(-h_i(-c_i - W_i v)\big)\right) \\ &= -b'v - \sum_{i=1}^{n} ln\left(\sum_{h_i} exp\big(h_i(c_i + W_i v)\big)\right). \end{aligned} \tag{5.4}$$

5.1.1Conditional Probability

The definition of conditional probability is about calculating the probability under some given condition. We can inspect of this concept, to calculate the probability of hidden units under the given observed variables.

In order to do that, we have first to define the probability distribution for the RBM. As we make use of the EBM, we can also make use of definition of the probability distribution in Eq. (3.1). Now we can define the conditional probability of hidden variables given by observed variables equal to

$$
\begin{aligned}
P(h|v) &= \frac{exp\big(-(-b'v - c'h - h'Wv)\big)}{\sum_{\tilde{h}} exp\left(-\big(-b'v - c'\tilde{h} - \tilde{h}'Wv\big)\right)} \\
&= \frac{exp(b'v + c'h + h'Wv)}{\sum_{\tilde{h}} exp\big(b'v + c'\tilde{h} + \tilde{h}'Wv\big)}. \qquad (5.5)
\end{aligned}
$$

The h is in a vector format, so that we can separate it into n elements to have a further simplification as

$$
\begin{aligned}
\frac{exp(b'v + c'h + h'Wv)}{\sum_{\tilde{h}} exp\big(b'v + c'\tilde{h} + \tilde{h}'Wv\big)} &= \frac{exp(b'v)\, exp(c'h + h'Wv)}{exp(b'v) \sum_{\tilde{h}} exp\big(c'\tilde{h} + \tilde{h}'Wv\big)} \\
&= \frac{\prod_{i=1}^{n} exp(c_i{}'h_i + h_i{}'W_i v)}{\prod_{i=1}^{n} \sum_{\widetilde{h_i}} exp\left(c_i{}'\widetilde{h_\imath} + \widetilde{h_\imath}{}'W_i v\right)} \\
&= \prod_{i=1}^{n} \frac{exp(c_i{}'h_i + h_i{}'W_i v)}{\sum_{\widetilde{h_i}} exp\left(c_i{}'\widetilde{h_\imath} + \widetilde{h_\imath}{}'W_i v\right)} \\
&= \prod_{i=1}^{n} \frac{exp\big(h_i(c_i + W_i v)\big)}{\sum_{\widetilde{h_i}} exp\left(\widetilde{h_\imath}(c_i + W_i v)\right)} \\
P(h|v) &= \prod_{i=1}^{n} P(h_i|v). \qquad (5.6)
\end{aligned}
$$

Also, we can reverse the operation, a similar derivation for *P(v|h)* is using the same conditional probability mechanism as

$$
\begin{aligned}
P(v|h) &= \frac{exp\big(-(-b'v - c'h - h'Wv)\big)}{\sum_{\tilde{v}} exp\big(-(-b'\tilde{v} - c'\mathrm{h} - h'W\tilde{v})\big)} \\
&= \frac{\prod_{j=1}^{m} exp\big(b'v_j + h'W_j v_j\big)}{\prod_{\mathrm{j}=1}^{m} \sum_{\widetilde{v_j}} exp\big(b'\widetilde{v_j} + h'W_j \widetilde{v_j}\big)}
\end{aligned}
$$

$$= \prod_{j=1}^{m} \frac{exp\left(\left(b_j + h'W_j\right)v_j\right)}{\sum_{\widetilde{v_j}} exp\left(\left(b_j + h'W_j\right)\widetilde{v_j}\right)}$$

$$= \prod_{j=1}^{m} P\left(v_j|h\right). \tag{5.7}$$

5.1.2RBM with Binary Units

Assume that our RBM is in binary nodes, which all the visible and hidden units are only have two states, one or zero. In addition, we can continue to extend our Eq. (5.6) and Eq. (5.7), where $h_i \in \{0, 1\}$ and $v_i \in \{0, 1\}$. Then, we will have the formula

$$P(h_i = 1|v) = \frac{exp\left(h_i(c_i + W_i v)\right)}{\sum_{\widetilde{h_\iota}\in\{0,1\}} exp\left(\widetilde{h_\iota}(c_i + W_i v)\right)}$$

$$= \frac{exp\left(1 * (c_i + W_i v)\right)}{exp\left(0 * (c_i + W_i v)\right) + exp\left(1 * (c_i + W_i v)\right)}$$

$$= \frac{exp(c_i + W_i v)}{1 + exp(c_i + W_i v)}$$

$$= sigm(c_i + W_i v), \tag{5.8}$$

$$P\left(v_j = 1|h\right) = \frac{exp\left(\left(b_j + h'W_j\right)v_j\right)}{\sum_{\widetilde{v_j}\in\{0,1\}} exp\left(\left(b_j + h'W_j\right)\widetilde{v_j}\right)}$$

$$= \frac{exp\left(b_j + h'W_j\right)}{1 + exp\left(b_j + h'W_j\right)}$$

$$= sigm\left(b_j + h'W_j\right). \tag{5.9}$$

Also, the Eq. (5.4) can also take the advantage of the binary units, get the simpler representation in make use of $h_i \in \{0, 1\}$ as

$$\mathcal{F}(v) = -b'v - \sum_{i=1}^{n} ln\left(\sum_{h_i\in\{0,1\}} exp\left(h_i(c_i + W_i v)\right)\right)$$

$$= -b'v - \sum_{i=1}^{n} ln\left(1 + exp(c_i + W_i v)\right). \tag{5.10}$$

5.1.3Gibbs Sampling

We have discussed a lot time about the hidden variables or the hidden units, but we do not talk about how we can get it. In fact, what we have to do is to sample the hidden units.

Assume the samples of $p(x)$ can be obtained by running a Markov chain to convergence, we can use the Gibbs sampling [12] as the transition operator.

Gibbs sample of the joint of N random variables $\mathrm{S} = (S_1, S_2 \ldots S_N)$ can be done by a sequence of N sampling sub-step of the form $S_i \sim p(S_i|S_{-i})$, where S_{-i} contains the N-1 other random variable in sample S excluding S_i.

For the RBM, the sample of set of variables includes the observed variables and hidden variables. However, there is no direct relation in between different unit of the observed variables, also the hidden variables. We can see the whole observed or hidden variables as a sampling block. Make use of the Gibbs sampling, use the hidden variables to sample the observed variables, also can use the observed variables to sample the hidden variables. According to Eq. (5.9 - 5.10), the Markov chain [13] (see Figure 3) can be written in the representation as

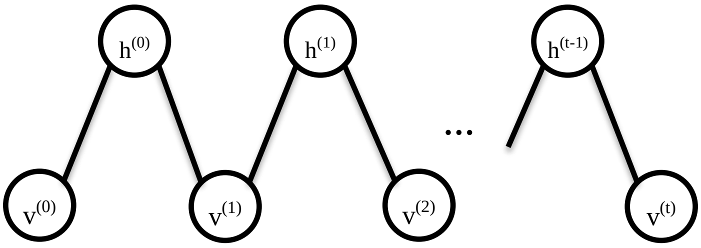


Figure 3: Markov Chain of Gibbs Sampling

$$h^{(t)} \sim P\left(h^{(t)} \middle| v^{(t)}\right) = sigm\left(Wv^{(t)} + c\right), \tag{5.11}$$

$$v^{(t)} \sim P\left(v^{(t)} | h^{(t-1)}\right) = sigm\left(W'h^{(t-1)} + b\right). \tag{5.12}$$

When t is large enough, our samples $(v^{(t)}, h^{(t)})$ can be very close to the accurate samples of p(v, h).

5.1.4Contrastive Divergence

In practical, the update of variables has to pass through the whole Markov chain is inefficiency. The other approach is to make use of the contrastive divergence (CD) [14], it is a method to do the sampling only in a number of steps, but not go through the whole chain. We denote one step of Gibbs sampling in the RBM is to sample both the observed variables and hidden variables one times. We make use of the CD-k [15] to represent the k-steps of Gibbs sampling. There is a lot of experiments show that the CD-k can have an acceptable result even when k equal to 1.

5.1.5Stochastic Gradient Descent in RBM

As we discuss before in CHAPTER 4, in order to optimize the EBM, we have to make use of the stochastic gradient descent method. We can apply the same technique in the RBM also. The update rule in Eq. (4.10) can only use the Gibbs sampling in CD-k to represent the expectations as

$$-\frac{\partial ln\left(P\left(v^i\right)\right)}{\partial \theta} \approx \frac{\partial \mathcal{F}\left(v^i\right)}{\partial \theta} - \frac{\partial \mathcal{F}\left(v^{(k)}\right)}{\partial \theta}. \tag{5.13}$$

5.2 Update RBM

According to RBM, there are three important variables in between it, which is the W, b and c, which all of them are the weights and biases for the node. We can make use of the Eq. (5.10), to calculate the partial derivatives of Free energy for each parameter

$$\begin{aligned}\frac{\partial\mathcal{F}(x)}{\partial W_{mn}} &= \frac{\partial\left(-b'v-\sum_n ln\left(1+exp(c_n+W_n v)\right)\right)}{\partial W_{mn}}\\ &= \frac{\partial\left(-ln\left(1+exp(c_n+W_n v)\right)\right)}{\partial W_{mn}}\\ &= -\frac{exp(c_n+W_n v)}{1+exp(c_n+W_n v)}\frac{\partial(c_n+W_n v)}{\partial W_{mn}}\\ &= -sigm(c_n+W_n v)*v_m, \qquad (5.14)\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{F}(x)}{\partial c_n} &= \frac{\partial\left(-b'v-\sum_n ln\left(1+exp(c_n+W_n v)\right)\right)}{\partial c_n}\\ &= \frac{\partial\left(-ln\left(1+exp(c_n+W_n v)\right)\right)}{\partial c_n}\\ &= -\frac{exp(c_n+W_n v)}{1+exp(c_n+W_n v)}\frac{\partial(c_n+W_n v)}{\partial c_n}\\ &= -sigm(c_n+W_n v), \qquad (5.15)\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{F}(x)}{\partial b_m} &= \frac{\partial\left(-b'v-\sum_n ln\left(1+exp(c_n+W_n v)\right)\right)}{\partial b_m}\\ &= \frac{\partial(-b'v)}{\partial b_m}\\ &= -v_m. \qquad (5.16)\end{aligned}$$

Combining the Eq. (5.13 - 5.16), we can get the log-likelihood gradients for the RBM as

$$\begin{aligned}-\frac{\partial ln\left(P\left(v^i\right)\right)}{\partial W_{mn}} &= \frac{\partial\mathcal{F}\left(v^i\right)}{\partial W_{mn}}-\frac{\partial\mathcal{F}\left(v^{(k)}\right)}{\partial W_{mn}}\\ &= -sigm\left(c_n+W_n v^i\right)*v_m^i+sigm\left(c_n+W_n v^{(k)}\right)*v_m^{(k)}, \qquad (5.17)\end{aligned}$$

$$\begin{aligned}-\frac{\partial ln\left(P\left(v^i\right)\right)}{\partial c_n} &= \frac{\partial\mathcal{F}\left(v^i\right)}{\partial c_n}-\frac{\partial\mathcal{F}\left(v^{(k)}\right)}{\partial c_n}\\ &= -sigm\left(c_n+W_n v^i\right)+sigm\left(c_n+W_n v^{(k)}\right), \qquad (5.18)\end{aligned}$$

$$\begin{aligned}
-\frac{\partial ln\left(P\left(v^{i}\right)\right)}{\partial b_m} &= \frac{\partial \mathcal{F}\left(v^{i}\right)}{\partial b_m} - \frac{\partial \mathcal{F}\left(v^{(k)}\right)}{\partial b_m} \\
&= -v_m^{i} + v_m^{(k)}.
\end{aligned} \tag{5.19}$$

CHAPTER 6. Experiment

6.1 Description

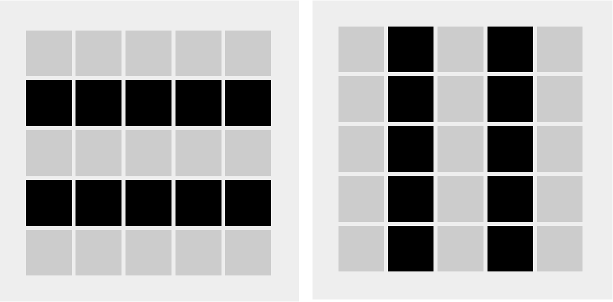


Figure 4: Bar-and-Stripe Benchmark

Our experiment is focus on the Bar-and-Stripe Benchmark (BAS) (see Figure 4), each of them has n × n units. Each of the benchmark will randomly choose a direction in horizontal or vertical. Then we will randomly generate a pattern for n values in binary. Assign the pattern into the whole horizontal or vertical lines in the benchmark.

Therefore, there are total $2 \times 2^n - 2$ different BAS benchmarks. We would like to use the RBM to learning probability for different benchmarks.

6.2 Unsupervised Learning

What we are doing is to train the RBM in the unsupervised learning method. We first generate sixty thousands of samples in Bar-and-Stripe Benchmark, and use the data to do ten epoch of training. The follow (see Table 1) is our result.

Table 1: Experiment Result for Different Dimension

Dimension	Num. Cases	Hidden Units	Training Time(ms)	Accuracy
3x3	14	100	7489	99.99% ±0.01%
4x4	30	100	9264	99.99% ±0.01%
5x5	62	100	10981	99.99% ±0.01%
6x6	126	100	13387	99.10% ±0.90%
7x7	254	100	16038	95.64% ±3.31%
8x8	510	100	20965	63.00% ±5.72%
9x9	1022	100	24093	26.05% ±4.18%
10x10	2046	100	26823	6.83% ±2.01%

From this table, we can see that when the dimension become larger, the number of cases will have an exponentially increase. The accuracy have an exponentially decrease (see Figure 5), and the training time increase smoothly (see Figure 6) according to the data length.

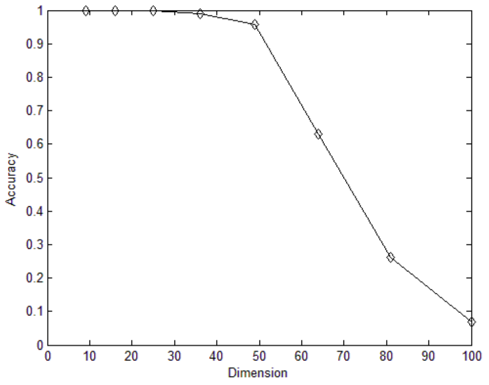


Figure 5: Accuracy Decrease when Dimension Increase

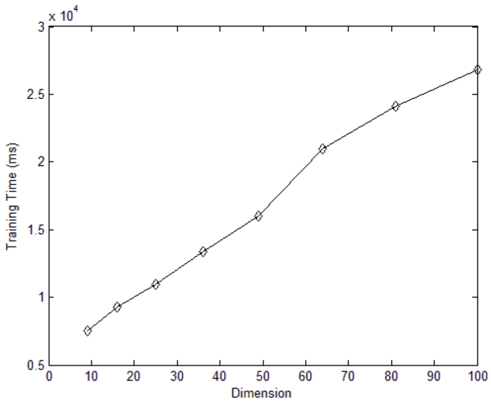


Figure 6: Time Complexity for Different Dimension

We can see that the accuracy decrease so fast against to the number of cases, but not the dimension. So that what we decide to do is to increase number of hidden unit, to increase the capability of learning of our model. We get the result is shown on (see Table 2).

Table 2: Experiment Result for Different Hidden Units

Dimension	Num. Cases	Hidden Units	Training Time(ms)	Accuracy
10x10	2046	100	26823	6.83% ±2.01%
10x10	2046	200	52859	55.79% ±4.34%
10x10	2046	300	79652	76.56% ±3.50%
10x10	2046	400	103688	87.68% ±3.19%
10x10	2046	500	130488	95.61% ±1.39%
10x10	2046	600	153691	97.07% ±1.22%
10x10	2046	700	179362	98.11% ±0.90%
10x10	2046	800	205010	93.60% ±1.10%
10x10	2046	900	232951	90.92% ±1.76%

10x10	2046	1000	254856	87.12% ±2.53%
10x10	2046	2000	516659	95.69% ±1.38%

From our experiment, we can find out that when the number of hidden units increases, the running time will also have the linear increase (see Figure 8). The accuracy will also increase until a certain point (see Figure 7), after that will decrease a little bit. What we consider is that, when the number of hidden units increase, the uncertainty factor will also increase correspondingly, which will cause this kind of problem. How to choose the number of hidden units is still a big problem in our point of view.

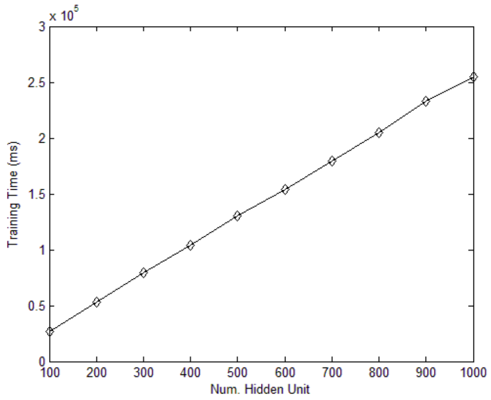


Figure 8: Training Time Linear Increase for Increase Hidden Units

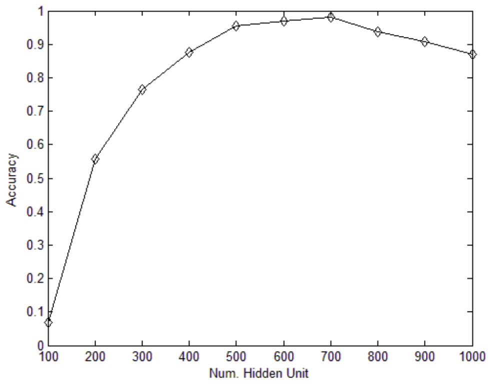


Figure 7: The Peak of Accuracy During The Number of Hidden Units Increase

6.3 Distribute Environment

6.3.1Introduction

Hadoop [16] is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users. And which is licensed under the Apache License 2.0.

6.3.2File System

The Hadoop distributed file system (HDFS) [17] is a distributed, scalable, and portable file-system written in Java for the Hadoop framework. The whole file system is a cluster of datanodes form the HDFS cluster (see Figure 9).

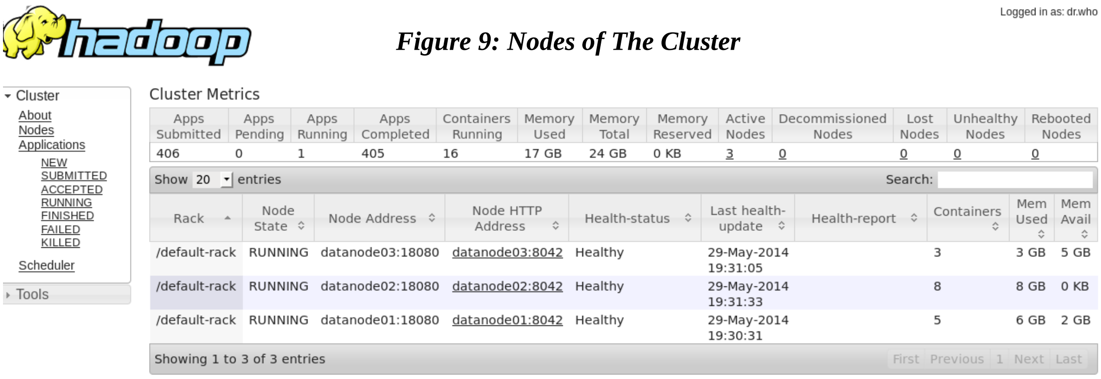


Figure 9: Nodes of The Cluster

Cluster Metrics

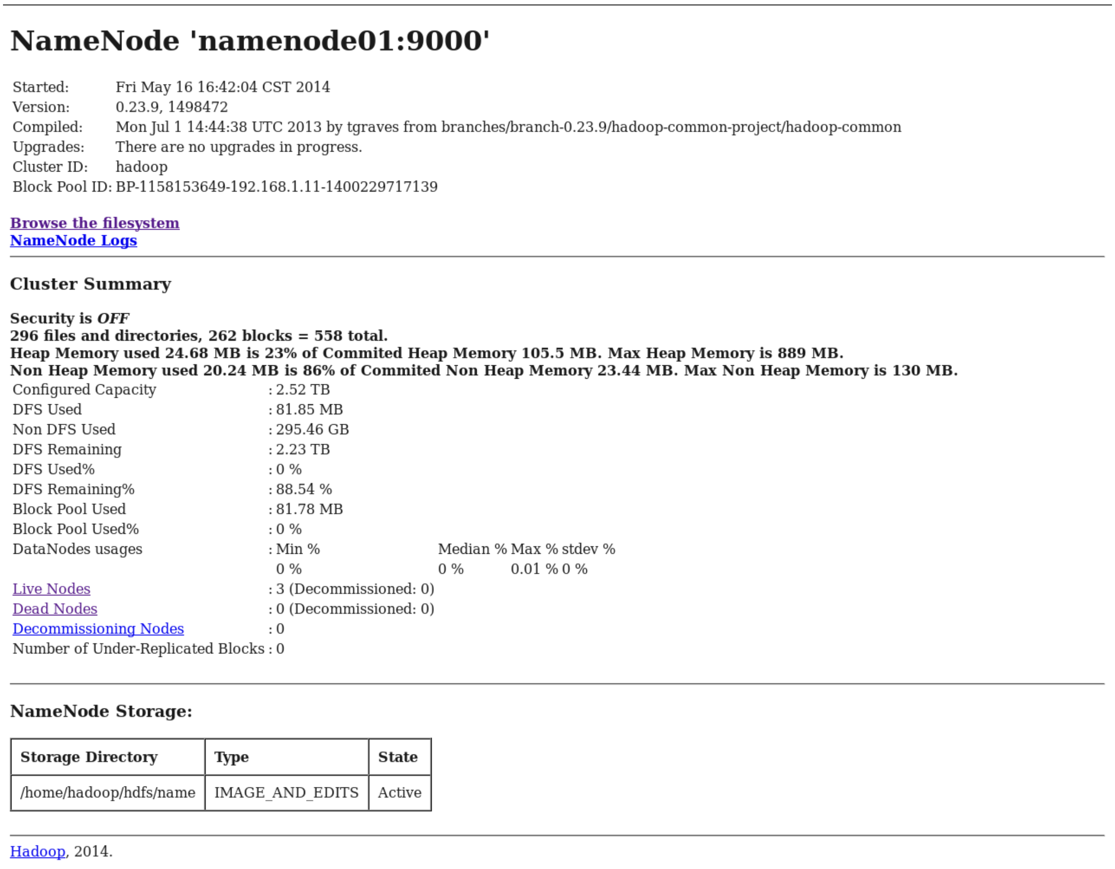
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
406	0	1	405	16	17 GB	24 GB	0 KB	3	0	0	0	0

Show 20 entries Search:

Rack	Node State	Node Address	Node HTTP Address	Health-status	Last health-update	Health-report	Containers	Mem Used	Mem Avail
/default-rack	RUNNING	datanode03:18080	datanode03:8042	Healthy	29-May-2014 19:31:05		3	3 GB	5 GB
/default-rack	RUNNING	datanode02:18080	datanode02:8042	Healthy	29-May-2014 19:31:33		8	8 GB	0 KB
/default-rack	RUNNING	datanode01:18080	datanode01:8042	Healthy	29-May-2014 19:30:31		5	6 GB	2 GB

Showing 1 to 3 of 3 entries

And also there is some important information about the File System (see Figure 10)



NameNode 'namenode01:9000'

Started:	Fri May 16 16:42:04 CST 2014
Version:	0.23.9, 1498472
Compiled:	Mon Jul 1 14:44:38 UTC 2013 by tgraves from branches/branch-0.23.9/hadoop-common-project/hadoop-common
Upgrades:	There are no upgrades in progress.
Cluster ID:	hadoop
Block Pool ID:	BP-1158153649-192.168.1.11-1400229717139

Browse the filesystem
NameNode Logs

Cluster Summary

Security is *OFF*
296 files and directories, 262 blocks = 558 total.
Heap Memory used 24.68 MB is 23% of Commited Heap Memory 105.5 MB. Max Heap Memory is 889 MB.
Non Heap Memory used 20.24 MB is 86% of Commited Non Heap Memory 23.44 MB. Max Non Heap Memory is 130 MB.

Configured Capacity	: 2.52 TB				
DFS Used	: 81.85 MB				
Non DFS Used	: 295.46 GB				
DFS Remaining	: 2.23 TB				
DFS Used%	: 0 %				
DFS Remaining%	: 88.54 %				
Block Pool Used	: 81.78 MB				
Block Pool Used%	: 0 %				
DataNodes usages	: Min %	Median %	Max %	stdev %	
	0 %	0 %	0.01 %	0 %	
Live Nodes	: 3 (Decommissioned: 0)				
Dead Nodes	: 0 (Decommissioned: 0)				
Decommissioning Nodes	: 0				
Number of Under-Replicated Blocks	: 0				

NameNode Storage:

Storage Directory	Type	State
/home/hadoop/hdfs/name	IMAGE_AND_EDITS	Active

Hadoop, 2014.

Figure 10: Information of HDFS

6.3.3Map Reduce Framework

Map Reduce is a framework [18] (see Figure 11) that can help us to handle the big data. First this framework will separate the input stream into different splits, then each split will assign to one of the datanode to do the map process. When the map process has finished, it will have some key value pair collect for the next step. During the collecting, the framework will help us to sort the data from all of the data nodes and according by the key, and the merge all the values belong to this key into a vector. The next is the reduce process, it will accept the key and value vector merged before. The reduce will also collect the data and which is the final output for this framework.

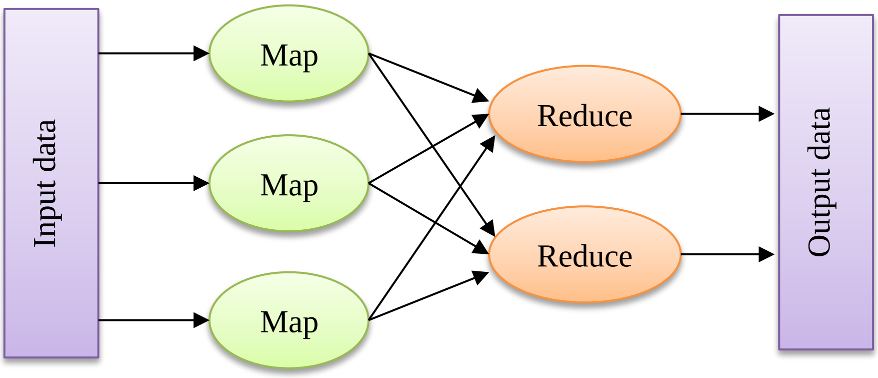


Figure 11: Map Reduce Framework

6.3.4Distribute Algorithm

What we found out that the most time consuming part is that it has to calculate a lot of matrix multiplication and sigmoid function during the update process of RBM, but it only use very few of time to update parameters. Therefore, we decide to put the matrix multiplication and the sigmoid calculation in parallel. The MR framework is a good tool for conducting the parallel compilation, and also the data collection.

The table (see Table 3) is the time costs during the MR training process. We can see that the CUP time increase when the number of splits increases. But at the same time the training time is decrease. It is because of that the CPU time is summary of all different splits, and most of the splits can run at the same time. In our cluster, there are around twenty MR processes are running at the same time.

Table 3: Time Consuming in Different MR Process

Splits	Map (ms)	Reduce (ms)	GC (ms)	CPU (ms)	Training Time (ms)
1	42730 ±12558	2253 ±158	451 ±70	46670 ±13070	408683
5	55922 ±4569	2216 ±79	744 ±128	48175 ±3525	207543
10	83853 ±12098	3689 ±1478	1300 ±344	56475 ±3885	167552
20	142566 ±5193	5469 ±143	2706 ±102	72760 ±3540	184984
Without MR	--	--	--	--	206251

From this table, we can find out a problem that more splits would not have more benefit in the MR framework. As it has to use more time to do the communication in between different nodes, the MR initialize time is inefficient, etc. Therefore, how to balance the value of number of splits also one challenge remaining.

But we can see that, when the number of data increases, there is a trend that the MR framework will become more efficient, since we have not used the whole calculation power in each split.

6.4 Results

We choose the 5x5 cases during the demonstration, since it is easy to understand and also can display more clearly. During the training process, we are using the supervised learning combined with the unsupervised learning. The most different of them is the label will involve during the training process or not, and what we do is to see the label as part of our data. The whole training process is in an unsupervised learning.

6.4.1Interface

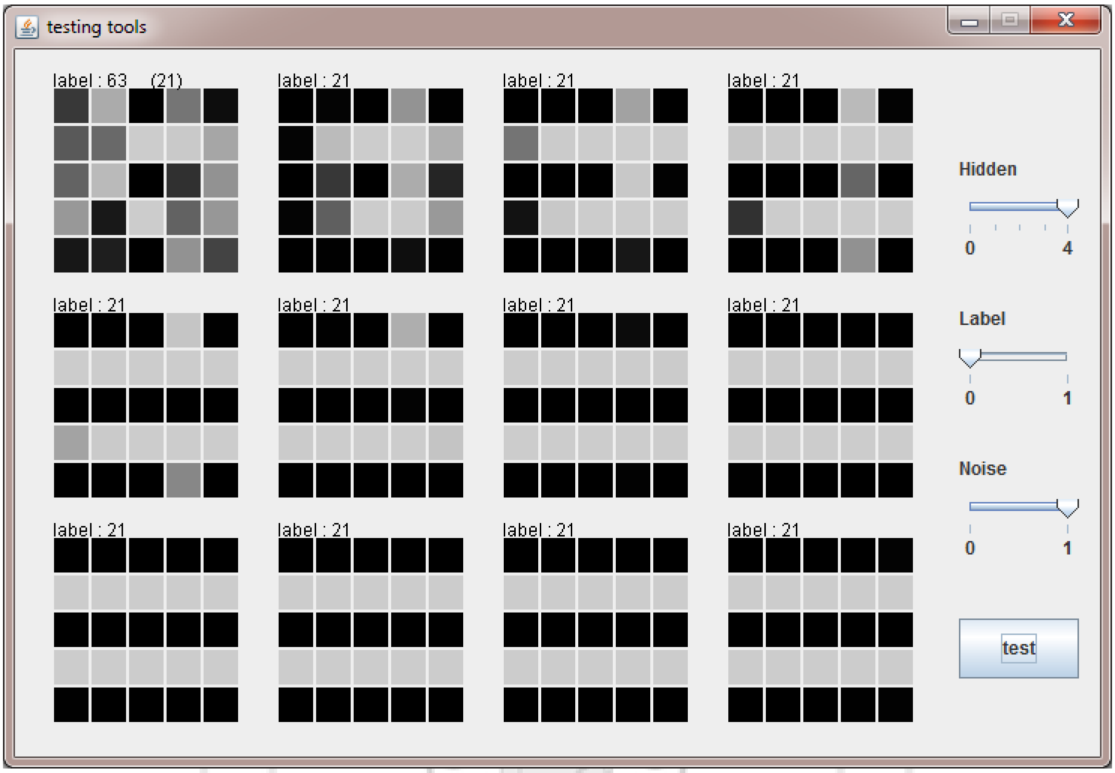


Figure 12: Interface of Our Testing Tools

In our interface (see Figure 12), we can see there are two part. There are twelve blocks contained in left hand side, each block has a label on top of it. The right hind side has some slider and a button there.

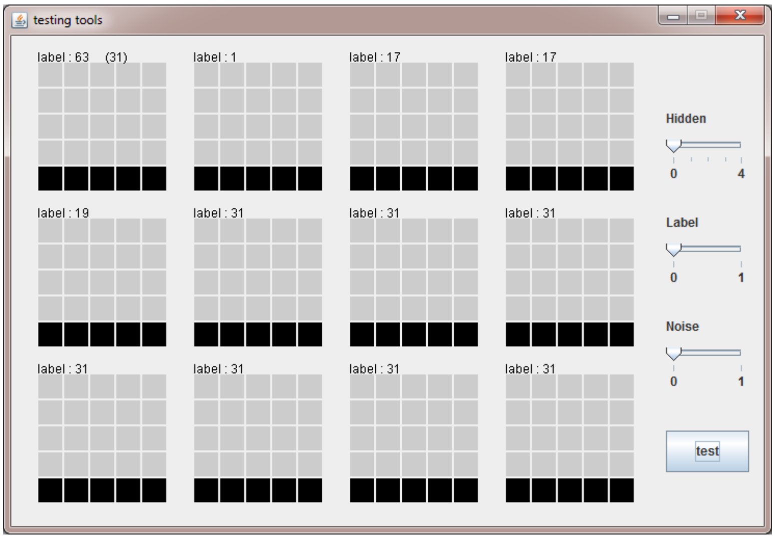
The block in the upper left corner is the data we want to pass into the model we trained. Beside the label, there is a bracket. Inside the bracket is the real label for the benchmark. From left to right, and up to down direction is in different Gibbs steps in the Gibbs sampling.

The first slider is to set how many random line of the benchmark will be hide. The second slider is to set will we given the real label in the sampling process, we define “1” to represent the label is given. The third slider is to set the disabling data. We use “0” as using the middle value of the data to replace it. When we use “1”, the disabling data will be replaced by some random variables. The button test is to take another testing.

6.4.2Classification

Classification problem [19] usually is a supervised learning problem, and we have transformed it into an unsupervised problem. We put the label into the data before the training process, which can help us to train the data and label simultaneously.

Our setting is very simple, since we want to classify which label this benchmark belong to. We just need to hide the label in the beginning of the sampling process.



label : 63 (31) label : 1 label : 17 label : 17

label : 19 label : 31 label : 31 label : 31

label : 31 label : 31 label : 31 label : 31

Figure 13: Experiment in Classification Problem

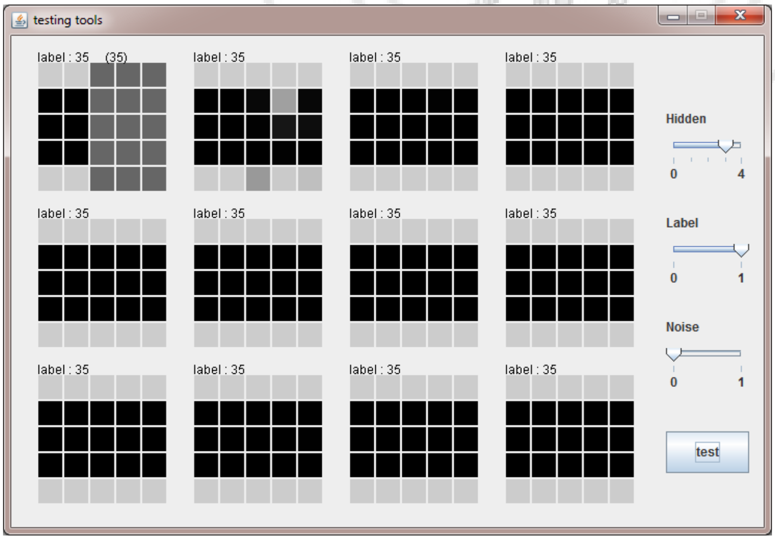
From the result (see Figure 13), we can see, since we only have 62 different cases, so we first assign a label with 63 to it, and the true label is inside the parentheses, and finally we can see it can learn the label very well.

From this experimental result, we can conclude that the RBM have the capability of handle the classification problem, and the classification accuracy is also acceptable.

6.4.3Image Restoration

Image Restoration problem is that a part of the image is lost, which may because of transmission or compression. We know what the object is, such as we know the label.

We can set there are part of the benchmark is lost, we try to use the model to restore the original benchmark.



label : 35 (35) label : 35 label : 35 label : 35

label : 35 label : 35 label : 35 label : 35

label : 35 label : 35 label : 35 label : 35

Figure 14: Experiment in Image Restoration

From the result (see Figure 14), we can see that the missing part can be almost implanted in the first step of the sample.

In this experiment, we can see that the RBM has the capability to learn the probability from some unknown data [20]. The image restoration is successful at all.

6.4.4Clustering

Some time we may get some data that we haven’t seen before, and we want to know what it most likely belongs to. According to some predesigned knowledge, such as the model what we trained, to clustering [21] it into one group of data. In addition, we use our benchmark as the cluster centre, and the label as the cluster group id.

We can use our testing tool to make the whole block become almost mess, just remain a little pattern belong to the BAS benchmark. We set there are four line of data is in a random data, only remain one line contain the pattern of BAS benchmark. At the same time, we disable the label in the beginning of the sampling process.

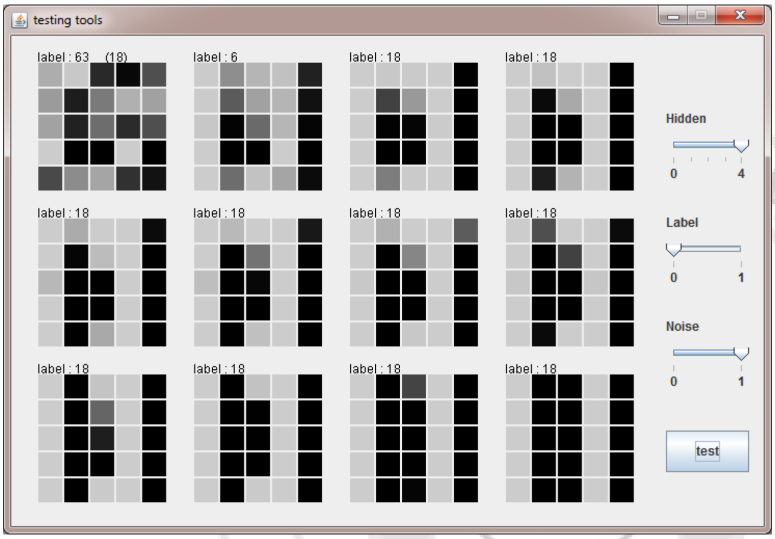


Figure 15: Experiment in Clustering Problem

From the result (see Figure 15), we firstly have to find out where is the pattern of BAS benchmark. We notice that the fourth row is what we want to find.

The sample is become clearer for more Gibbs steps, which can learn the cluster group number as our expectation for this sample.

In this experiment, we can see that the mess data become closer to one of our cluster centre and cluster group in the Gibbs sampling. Finally we find out that data and label is same as we generated. We think the RBM is good for clustering.

However, we mess up almost all the data, the samples which generated from the model may not be the same as what our expectation. In the clustering process may directly go to another cluster different from our expectation is acceptable.

CHAPTER 7. Conclusion

In our report, we discuss the process of how to train the RBM, the performance of RBM in different settings, and the distributed implementation.

Also, we meet some problem during our research, like how to choose the number of hidden variables for the RBM, how to choose the number of splits in the distributed environment.

During the experiments, we found out the RBM can learn both vertical and horizontal features at the same time (see Figure 16). Luckily, we find out the sample will most likely converge to our expectation when we do more step of the sampling (see Figure 17).

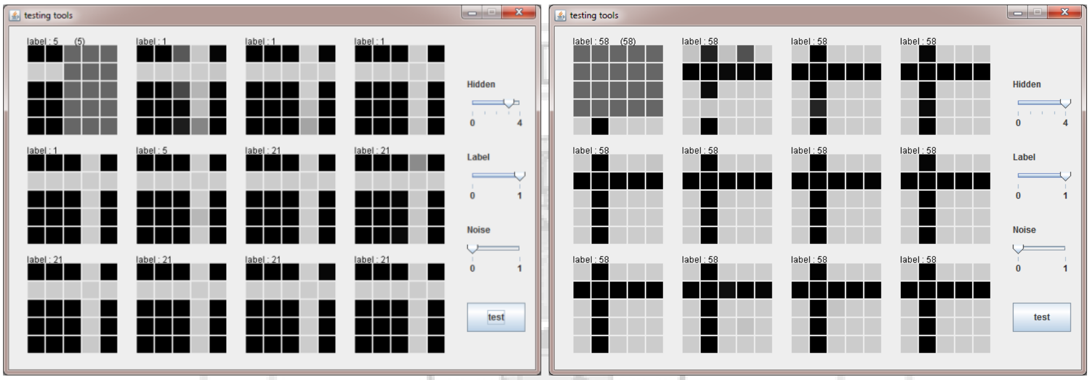


Figure 16: Learning both Vertical and Horizontal Features

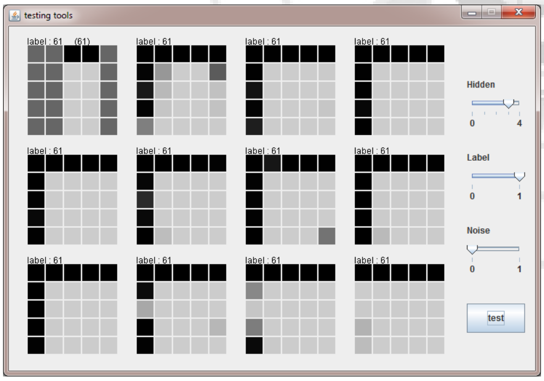


Figure 17: Automatically Correct in Gibbs Sampling

When the dimension of the data increases, we have to use more hidden units to guarantee that the accuracy is acceptable in the sampling. There is a linear increase in training time when the dimension of the data increases linearly, also a linear increase with respect to the number of hidden variables. Combining these two factors, we can see the trend that the training time will have a significant increase.

Distributed algorithm and parallel calculation obviously can help us to improve the performance in training the model. How to improve the distributed algorithm, increase the utilization of all the nodes inside the cluster and decrease the training time are still big challenges for us.

CHAPTER 8. References

[1] Bengio, Yoshua. "Learning deep architectures for AI." Foundations and trends® in Machine Learning 2.1 (2009): 1-127.

[2] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.

[3] Salakhutdinov, Ruslan, and Geoffrey Hinton. "An efficient learning procedure for deep Boltzmann machines." Neural computation 24.8 (2012): 1967-2006.

[4] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." Science 313.5786 (2006): 504-507.

[5] Le Roux, Nicolas, and Yoshua Bengio. "Representational power of restricted Boltzmann machines and deep belief networks." *Neural Computation* 20.6 (2008): 1631-1649.

[6] Hinton, Geoffrey. "A practical guide to training restricted Boltzmann machines." Momentum 9.1 (2010): 926.

[7] Bengio, Yoshua, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives." (2013): 1-1.

[8] Bottou, Léon. "Large-scale machine learning with stochastic gradient descent" *Proceedings of COMPSTAT'2010*. Physica-Verlag HD, 2010. 177-186.

[9] Zinkevich, Martin, et al. "Parallelized Stochastic Gradient Descent." *NIPS*. Vol. 4. No. 1. 2010.

[10] Salakhutdinov, Ruslan. "Learning in Markov Random Fields using Tempered Transitions" *NIPS*. Vol. 22. 2009.

[11] Hinton, Geoffrey E. "Learning multiple layers of representation." *Trends in cognitive sciences* 11.10 (2007): 428-434.

[12] Desjardins, Guillaume, et al. "Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines." *International Conference on Artificial Intelligence and Statistics*. 2010.

[13] Bengio, Yoshua, et al. "Greedy layer-wise training of deep networks." *Advances in neural information processing systems* 19 (2007): 153.

[14] Carreira-Perpinan, Miguel A., and Geoffrey E. Hinton. "On contrastive divergence learning." Proceedings of the tenth international workshop on artificial intelligence and statistics. NP: Society for Artificial Intelligence and Statistics, 2005.

[15] Tieleman, Tijmen. "Training restricted Boltzmann machines using approximations to the likelihood gradient." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

[16] White, Tom. *Hadoop: The Definitive Guide: The Definitive Guide*. O'Reilly Media, 2009.

[17] Shvachko, Konstantin, et al. "The hadoop distributed file system." *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*. IEEE, 2010.

[18] Chu, Cheng-Tao, et al. "Map-reduce for machine learning on multicore." *NIPS*. Vol. 6. 2006.

[19] Larochelle, Hugo, and Yoshua Bengio. "Classification using discriminative restricted Boltzmann machines." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

[20] Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." *Proceedings of the 24th international conference on Machine learning*. ACM, 2007.

[21] Coates, Adam, Andrew Y. Ng, and Honglak Lee. "An analysis of single-layer networks in unsupervised feature learning." *International Conference on Artificial Intelligence and Statistics*. 2011.

CHAPTER 9. Appendix

Cluster: Apache™ Hadoop®

Version: Hadoop 0.23.9

Num. Namenode: 1

Num. Datanode: 3

Operating System: CentOS release 6.4 (Final)

Kernel: Linux Version 2.6.32-358.el6.x86_64

Java: Java™ SE Runtime Environment (build 1.7.0_40-b43)

Computers: HP Compaq Elite 8300 MT PC

Processor: Intel® Core™ i7-3770 CPU @ 3.40GHz

RAM: 8 GB

NIC: Intel® PRO/1000 Network Connection

Devices: NETGEAR N600 Wireless Dual Band Gigabit Router WNDR3700v2

Num. port: 4

NIC: Gigabit Ethernet