Scientific Research Publishing

Advances in Pure Mathematics, 2018, 8, 307-320
http://www.scirp.org/journal/apm
ISSN Online: 2160-0384
ISSN Print: 2160-0368

Mathematical Reinforcement to the Minibatch of Deep Learning

Kazuyuki Fujii*

International College of Arts and Sciences, Yokohama City University, Yokohama, Japan
Email: fujii@yokohama-cu.ac.jp

How to cite this paper: Fujii, K. (2018) Mathematical Reinforcement to the Minibatch of Deep Learning. *Advances in Pure Mathematics*, **8**, 307-320.
https://doi.org/10.4236/apm.2018.83016

Received: February 13, 2018
Accepted: March 20, 2018
Published: March 23, 2018

Abstract

We elucidate a practical method in Deep Learning called the minibatch which is very useful to avoid local minima. The mathematical structure of this method is, however, a bit obscure. We emphasize that a certain condition, which is not explicitly stated in ordinary expositions, is essential for the minibatch method. We present a comprehensive description Deep Learning for non-experts with the mathematical reinforcement.

Keywords

Deep Learning, Minibatch, Error (Loss) Function, Local Minima

1. Introduction

Deep Learning is one of Machine Learning methods and it has attracted much attention recently. Leaning in general is divided into the supervised learning and unsupervised learning. In this paper we discuss only supervised learning. For general introduction to Deep Learning, see for example [1]. The monographs [2] [3] [4] are standard textbooks.

Deep Learning is based on big data, so the supervised learning will give a heavy load to the computer. In order to alleviate the burden we usually use a practical method called the minibatch (a collection of randomly selected small data from big data, see **Figure 5**). Although the method is commonly used it is not widely understood why it is so effective from the mathematical viewpoint. We point out in Theorem II that a certain condition, which is practically satisfied in ordinary applications, is essential for the effectiveness of the minibatch method.

In this paper we present a short and concise review of Deep Learning for

*Dedicated to the memory of Professor Ichiro Yokota.

non-experts and provide a mathematical reinforcement to the minibatch from the viewpoint of Linear Algebra. Theorem I and II in the text are our main results and some related problems are presented.

After reading this paper, readers are encouraged to tackle a remarkable paper [5] which is definitely a monumental achievement.

2. Simple Neural Network

As a general introduction to Neuron Dynamics see for example [6] [7].

For non-experts let us draw a neuron model based on the step function $S(x)$. In **Figure 1** the set $\{x_1, x_2, \cdots, x_n\}$ is the input data, $z = S(y)$ is the output data and the set $\{w_1, w_2, \cdots, w_n\}$ is the weights of synaptic connections. The parameter θ is the threshold of the neuron.

Here, $z = S(x)$ is a simple step function defined by

$$S(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$$

Since an algorithm called the backpropagation in Neural Network System uses derivatives of some activation functions, the simple step function $S(x)$ is not suitable. Therefore, we usually use a function called the sigmoid function instead of the step function (see **Figure 2**). This function is used for a nonlinear compression of data.

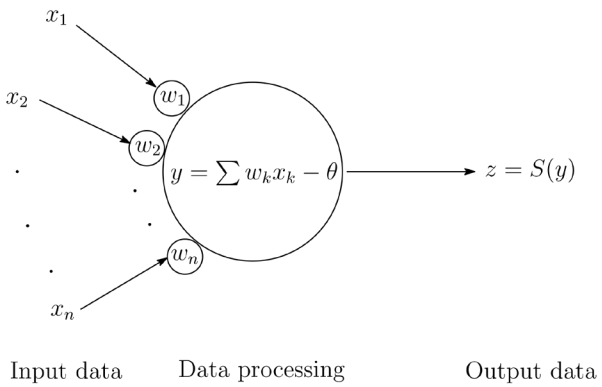

Figure 1. Simple neuron model.

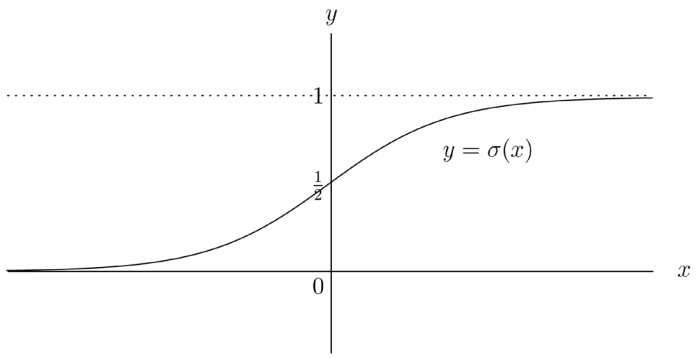

Figure 2. Sigmoid function.

Definition The sigmoid function is defined by

$$\sigma(x)=\frac{1}{1+\mathrm{e}^{-x}} \quad (x\in\mathbf{R}). \tag{1}$$

It is easy to see that the function satisfies

$$\sigma(-\infty)=0,\ \sigma(0)=\frac{1}{2},\ \sigma(\infty)=1$$

and its derivatives are expressed as polynomials of the original function

$$\begin{aligned}&\sigma'(x)=\sigma(x)\big(1-\sigma(x)\big),\\&\sigma''(x)=\sigma(x)\big(1-\sigma(x)\big)\big(1-2\sigma(x)\big).\end{aligned} \tag{2}$$

The graph is as follows.

Comment In Mathematics this is a famous function called the standard logistic function, while in Deep Learning this is called the sigmoid function.

A general logistic function $f(x)$ has three parameters $\{L,\lambda,x_0\}$, the overall height L, the parameter λ specifying the inverse width of the transient region and the location of the center x_0:

$$f(x)=\frac{L}{1+\mathrm{e}^{-\lambda(x-x_0)}}.$$

The standard logistic function is characterized by $L=1,\lambda=1,x_0=0$. See a lecture note [8] as to why the function is so important.

Let us draw a neuron model based on the sigmoid function, See **Figure 3**.

The most important thing is to improve the set of weights of the synoptic connections $\{w_1,w_2,\cdots,w_n\}$ by hard learning. For the purpose we prepare m input data and teacher signals. In the following we assume $m<n$ and $\theta=0$ for simplicity. This causes no problem for the present explanation. There is a suitable value of θ for each specific application.

For $j=1,2,\cdots,m$ we set

$$\text{Input data}:\boldsymbol{x}^{(j)}=\left(x_1^{(j)},x_2^{(j)},\cdots,x_n^{(j)}\right)^{\mathrm{T}},$$

$$\text{Teacher signal}:d^{(j)}$$

where T denotes the transpose of a vector (or a matrix).

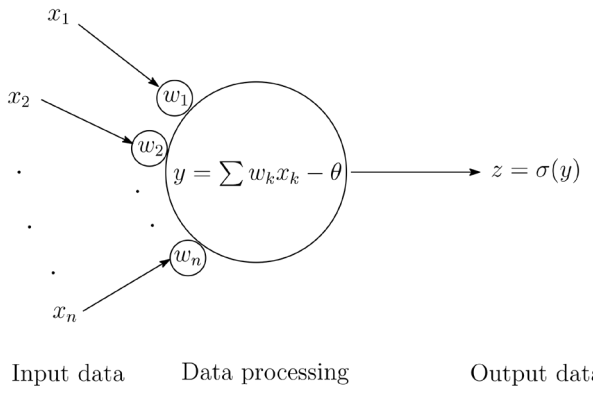

Figure 3. Simple neuron model modified.

Our method is to determine the weights of synaptic connections which match the output data with given teacher signals as much as possible. For the purpose we consider a square error like

$$E = E(\boldsymbol{w}) = \frac{1}{2}\sum_{j=1}^{m}\left(d^{(j)} - z^{(j)}\right)^2, \tag{3}$$

$$z^{(j)} = \sigma\left(y^{(j)}\right) = \sigma\left(\sum_{k=1}^{n} w_k x_k^{(j)}\right) \tag{4}$$

and determine $\boldsymbol{w} = (w_1, w_2, \cdots, w_n)^{\mathrm{T}}$ so that $E(\boldsymbol{w})$ is minimized.

We want to realize this purpose by repeated learning. For that we consider a discrete dynamical system. Namely, we set

$$E = E(\boldsymbol{w}(t)) = \frac{1}{2}\sum_{j=1}^{m}\left(d^{(j)} - z^{(j)}(t)\right)^2;\ z^{(j)}(t) = \sigma\left(y^{(j)}(t)\right) = \sigma\left(\sum_{k=1}^{n} w_k(t) x_k^{(j)}\right) \tag{5}$$

for $t = 0,1,2,\cdots,T\ (\in \mathbf{N})$.

The initial value $\boldsymbol{w}(0)$ is given appropriately (not so important).

2.1. Gradient Descent

For a general introduction to Gradient Descent see for example [9].

The gradient descent is performed by changing $w_k(t)$ to

$$w_k(t+1) = w_k(t) - \epsilon\frac{\partial E(\boldsymbol{w}(t))}{\partial w_k(t)} \quad (0 < \epsilon < 1) \tag{6}$$

and we evaluate the error with the renewed weights

$$E = E(\boldsymbol{w}(t+1)).$$

This renewal of the weights is the learning process. This is very popular in Mathematics[1]. We repeat this procedure T steps which is large enough like

$$E(\boldsymbol{w}(0)) \Rightarrow E(\boldsymbol{w}(1)) \Rightarrow \cdots \Rightarrow E(\boldsymbol{w}(T)) \equiv E(\boldsymbol{w}).$$

Let us proceed. We set $E = E(\boldsymbol{w})$ for simplicity (t omitted). From (5) simple calculation gives

$$\begin{aligned}\frac{\partial E}{\partial w_k} &= \sum_{j=1}^{m}\left(d^{(j)} - z^{(j)}\right)\left(-\frac{\partial z^{(j)}}{\partial w_k}\right)\\ &= -\sum_{j=1}^{m}\left(d^{(j)} - z^{(j)}\right)\sigma\left(\sum_{k=1}^{n} w_k x_k^{(j)}\right)\left\{1 - \sigma\left(\sum_{k=1}^{n} w_k x_k^{(j)}\right)\right\} x_k^{(j)}\\ &= -\sum_{j=1}^{m}\left(d^{(j)} - z^{(j)}\right) x_k^{(j)}\sigma\left(\sum_{k=1}^{n} w_k x_k^{(j)}\right)\left\{1 - \sigma\left(\sum_{k=1}^{n} w_k x_k^{(j)}\right)\right\}\end{aligned}$$

and from (6) we obtain

$$\begin{aligned}w_k(t+1) &= w_k(t) - \epsilon\frac{\partial E(w(t))}{\partial w_k(t)}\\ &= w_k(t) + \epsilon\sum_{j=1}^{m}\left(d^{(j)} - z^{(j)}(t)\right) x_k^{(j)}\sigma\left(\sum_{k=1}^{n} w_k(t) x_k^{(j)}\right)\left\{1 - \sigma\left(\sum_{k=1}^{n} w_k(t) x_k^{(j)}\right)\right\}.\end{aligned}$$

[1]However, to choose ϵ in (6) correctly is a very hard problem.

By substituting this into (5) we obtain

$$z^{(j)}(t+1)=\sigma\left(\sum_{k=1}^{n}w_k(t+1)x_k^{(j)}\right)$$

and the square error becomes

$$E=E(\boldsymbol{w}(t+1))=\frac{1}{2}\sum_{j=1}^{m}\left(d^{(j)}-z^{(j)}(t+1)\right)^2,$$

which is complicated enough. The renewal of the weights terminates when all $\partial E(\boldsymbol{w})/\partial w_k=0$, which is a critical point as shown in the next subsection.

2.2. Number of Critical Points

Here we evaluate the number of critical points to be encountered in the process. First of all, let us show the definition of a critical point with a simple example.

Definition Let $f=f(x,y,z)$ be a smooth function. Then a critical (or stationary) point (x_0,y_0,z_0) is defined by

$$\begin{cases} f_x(x_0,y_0,z_0)=0, \\ f_y(x_0,y_0,z_0)=0, \\ f_z(x_0,y_0,z_0)=0. \end{cases} \tag{7}$$

A critical point is not necessarily a point giving local minimum or local maximum.

We make an important assumption with respect to the input data.

Assumption (good data) For $m<n$

$$\left\{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\cdots,\boldsymbol{x}^{(m)}\right\} \tag{8}$$

are linearly independent.

The assumption tells that there is a set of integers $k_1<k_2<\cdots<k_m$ satisfying

$$\begin{vmatrix} x_{k_1}^{(1)} & x_{k_1}^{(2)} & \cdots & x_{k_1}^{(m)} \\ x_{k_2}^{(1)} & x_{k_2}^{(2)} & \cdots & x_{k_2}^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k_m}^{(1)} & x_{k_m}^{(2)} & \cdots & x_{k_m}^{(m)} \end{vmatrix}\neq 0. \tag{9}$$

Let us recall

$$E=\frac{1}{2}\sum_{j=1}^{m}\left(d^{(j)}-z^{(j)}\right)^2=\frac{1}{2}\sum_{j=1}^{m}\left(d^{(j)}-\sigma\left(y^{(j)}\right)\right)^2,\quad \sigma\left(y^{(j)}\right)=\sigma\left(\sum_{k=1}^{n}w_kx_k^{(j)}\right)$$

and try to find if E has a critical point. Suppose E has a critical point at the weights $\boldsymbol{w}=\{w_1,\cdots,w_n\}^{\mathrm{T}}$,

$$0=\frac{\partial E}{\partial w_k}=-\sum_{j=1}^{m}\left(d^{(j)}-z^{(j)}\right)\sigma'\left(y^{(j)}\right)x_k^{(j)},\quad k=1,\cdots,n.$$

Let us set

$$A^{(j)}=\left(d^{(j)}-z^{(j)}\right)\sigma'\left(y^{(j)}\right)$$

for simplicity. The equation can be written as

$$\sum_{j=1}^{m} A^{(j)} x_k^{(j)} = 0 \quad \text{for } k = 1, \cdots, n$$

and it implies

$$\sum_{j=1}^{m} x_k^{(j)} A^{(j)} = 0 \quad \text{for } k = k_1, k_2, \cdots, k_m$$

or in matrix form

$$\begin{pmatrix} x_{k_1}^{(1)} & x_{k_1}^{(2)} & \cdots & x_{k_1}^{(m)} \\ x_{k_2}^{(1)} & x_{k_2}^{(2)} & \cdots & x_{k_2}^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k_m}^{(1)} & x_{k_m}^{(2)} & \cdots & x_{k_m}^{(m)} \end{pmatrix} \begin{pmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(m)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Since the determinant of coefficient matrix is non-zero from (9) we have

$$\begin{pmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(m)} \end{pmatrix} = \begin{pmatrix} \left(d^{(1)} - z^{(1)}\right)\sigma'\left(y^{(1)}\right) \\ \left(d^{(2)} - z^{(2)}\right)\sigma'\left(y^{(2)}\right) \\ \vdots \\ \left(d^{(m)} - z^{(m)}\right)\sigma'\left(y^{(m)}\right) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

and

$$d^{(j)} - z^{(j)} = d^{(j)} - \sigma\left(y^{(j)}\right) = 0 \quad \text{for } j = 1, 2, \cdots, m$$

because $\sigma'(y) \neq 0$. This means

$$E = \frac{1}{2}\sum_{j=1}^{m}\left(d^{(j)} - z^{(j)}\right)^2 = 0.$$

The result shows that there is no critical point in this process except for $E = 0$. Therefore, when modifying $w_k(t)$ by the gradient descent (6) successively there is no danger of being trapped by points taking local minima. Let us summarize the result :

Theorem I Under the assumption (8) there is only one global minimum $E = 0$.

Next, let us explain how to choose linearly independent data. We assume that n is huge and m is sufficiently smaller than n.

For m input data

$$\left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(m)}\right\}$$

we consider the $m \times m$ Gram's deterninant

$$G\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(m)}\right) = \begin{vmatrix} \left\langle \boldsymbol{x}^{(1)} \middle| \boldsymbol{x}^{(1)} \right\rangle & \left\langle \boldsymbol{x}^{(1)} \middle| \boldsymbol{x}^{(2)} \right\rangle & \cdots & \left\langle \boldsymbol{x}^{(1)} \middle| \boldsymbol{x}^{(m)} \right\rangle \\ \left\langle \boldsymbol{x}^{(2)} \middle| \boldsymbol{x}^{(1)} \right\rangle & \left\langle \boldsymbol{x}^{(2)} \middle| \boldsymbol{x}^{(2)} \right\rangle & \cdots & \left\langle \boldsymbol{x}^{(2)} \middle| \boldsymbol{x}^{(m)} \right\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \left\langle \boldsymbol{x}^{(m)} \middle| \boldsymbol{x}^{(1)} \right\rangle & \left\langle \boldsymbol{x}^{(m)} \middle| \boldsymbol{x}^{(2)} \right\rangle & \cdots & \left\langle \boldsymbol{x}^{(m)} \middle| \boldsymbol{x}^{(m)} \right\rangle \end{vmatrix}. \quad (10)$$

If m is not large, evaluation of the Gram's determinant is not so difficult. It is well-known that the input data are linearly independent if the Gram's determinant is non-zero (see for example [10]).

Let us give a short explanation for non-experts. For simplicity we consider the case of $m=3$ and set

$$\boldsymbol{x}^{(1)}=\boldsymbol{a},\ \boldsymbol{x}^{(2)}=\boldsymbol{b},\ \boldsymbol{x}^{(3)}=\boldsymbol{c}.$$

Then

$$G(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})=\begin{vmatrix}\langle\boldsymbol{a}|\boldsymbol{a}\rangle & \langle\boldsymbol{a}|\boldsymbol{b}\rangle & \langle\boldsymbol{a}|\boldsymbol{c}\rangle\\ \langle\boldsymbol{b}|\boldsymbol{a}\rangle & \langle\boldsymbol{b}|\boldsymbol{b}\rangle & \langle\boldsymbol{b}|\boldsymbol{c}\rangle\\ \langle\boldsymbol{c}|\boldsymbol{a}\rangle & \langle\boldsymbol{c}|\boldsymbol{b}\rangle & \langle\boldsymbol{c}|\boldsymbol{c}\rangle\end{vmatrix}$$

and we assume that $\{\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}\}$ are linearly dependent. For example, we can set

$$\boldsymbol{c}=\alpha\boldsymbol{a}+\beta\boldsymbol{b}\quad(\alpha,\beta\text{ is real}).$$

Short calculation after substituting this into the Gram's determinant above gives

$$G(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})=\begin{vmatrix}\langle\boldsymbol{a}|\boldsymbol{a}\rangle & \langle\boldsymbol{a}|\boldsymbol{b}\rangle & \langle\boldsymbol{a}|\boldsymbol{c}\rangle\\ \langle\boldsymbol{b}|\boldsymbol{a}\rangle & \langle\boldsymbol{b}|\boldsymbol{b}\rangle & \langle\boldsymbol{b}|\boldsymbol{c}\rangle\\ \alpha\langle\boldsymbol{a}|\boldsymbol{a}\rangle+\beta\langle\boldsymbol{b}|\boldsymbol{a}\rangle & \alpha\langle\boldsymbol{a}|\boldsymbol{b}\rangle+\beta\langle\boldsymbol{b}|\boldsymbol{b}\rangle & \alpha\langle\boldsymbol{a}|\boldsymbol{c}\rangle+\beta\langle\boldsymbol{b}|\boldsymbol{c}\rangle\end{vmatrix}$$

and some fundamental operations of determinant give

$$G(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})=\begin{vmatrix}\langle\boldsymbol{a}|\boldsymbol{a}\rangle & \langle\boldsymbol{a}|\boldsymbol{b}\rangle & \langle\boldsymbol{a}|\boldsymbol{c}\rangle\\ \langle\boldsymbol{b}|\boldsymbol{a}\rangle & \langle\boldsymbol{b}|\boldsymbol{b}\rangle & \langle\boldsymbol{b}|\boldsymbol{c}\rangle\\ 0 & 0 & 0\end{vmatrix}=0.$$

Namely, we have shown

$$\{\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}\}\text{ is linearly dependent}\Rightarrow G(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})=0.$$

Taking the contraposition gives

$$G(\boldsymbol{a},\boldsymbol{b},\boldsymbol{c})\neq 0\Rightarrow\{\boldsymbol{a},\boldsymbol{b},\boldsymbol{c}\}\text{ is linearly independent.}$$

Comment For $n>m$ we denote by $M(n,m;\mathbf{R})$ a set of all $n\times m$ matrices over $\mathbf{R}$ and denote

$$M_{n,m}(\mathbf{R})=\left\{\left(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\cdots,\boldsymbol{x}^{(m)}\right)\in M(n,m;\mathbf{R})\,\middle|\,\left\{\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)},\cdots,\boldsymbol{x}^{(m)}\right\}\text{ is linearly independent}\right\}.$$

It is well-known that the subset $M_{n,m}(\mathbf{R})$ is open and dense in $M(n,m;\mathbf{R})$. Therefore, if we choose m vectors from $\mathbf{R}^n$ randomly they are almost all linearly independent.

3. Deep Neural Network

Deep Learning is the heart of Artificial Intelligence (AI) and we provide a concise review for non-experts. Deep Learning is a deep neural network consisting of the input layer (K components), the (multiplex) intermediate layers

($M_1, M_2, \cdots, N$ components and the output layer (N components). See Figure 4 (in which $M_1 = L, M_2 = M, M_3 = N$).

3.1. Backpropagation

First, let us show a figure of deep neural network[2] (the figure is short in intermediate layers).

Let us explain some notations.

$$\boldsymbol{x} = \left(x_1, x_2, \cdots, x_K\right)^{\mathrm{T}}$$

is the input data. In this case, the weights of synaptic connections are represented by matrices U, V, W given by

$$\begin{aligned} U &= \left(u_{ij}\right) \in M\left(L, K; \mathbf{R}\right), \\ V &= \left(v_{ij}\right) \in M\left(M, L; \mathbf{R}\right), \\ W &= \left(w_{ij}\right) \in M\left(N, M; \mathbf{R}\right) \end{aligned} \tag{11}$$

and our aim is educating these matrices.

The synaptic connections are expressed as

$$r_i = \sum_{j=1}^{M} w_{ij}\beta_j - \theta, \quad i = 1, \cdots, N$$

$$z_i = \sigma\left(r_i\right)$$

and

$$q_i = \sum_{j=1}^{L} v_{ij}\alpha_j - \theta, \quad i = 1, \cdots, M$$

$$\beta_i = \sigma\left(q_i\right)$$

and

$$p_i = \sum_{j=1}^{K} u_{ij}x_j - \theta, \quad i = 1, \cdots, L$$

$$\alpha_i = \sigma\left(p_i\right)$$

respectively. See Figure 4 once more.

A comment is in order. The flow of data is as follows:

$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix} \Rightarrow \boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_L \end{pmatrix} \Rightarrow \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{pmatrix} \Rightarrow \boldsymbol{z} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{pmatrix}. \tag{12}$$

Now, if

$$\boldsymbol{d} = \left(d_1, d_2, \cdots, d_N\right)^{\mathrm{T}}$$

is a set of teacher signals, then the square error becomes

[2]Figure 4 is of course not perfect because drawing a figure of a big size is not easy.

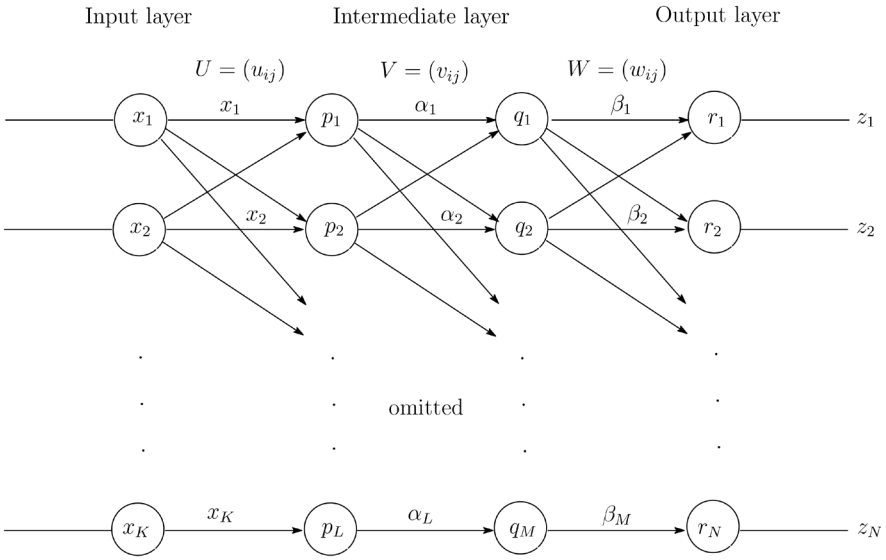

Figure 4. Deep neural network model.

$$E = E(U,V,W) = \frac{1}{2}\sum_{k=1}^{N}(d_k - z_k)^2.$$

Let us consider (discrete) time evolution of the model. For $t = 0,1,2,\cdots$, we must determine

$$t \to t+1$$

$$U(t) \to U(t+1),$$

$$V(t) \to V(t+1),$$

$$W(t) \to W(t+1),$$

$$E(t) \to E(t+1)$$

and we expect

$$E(t) \approx 0$$

if t is large enough (the end of Learning !).

Assume now that the system is in the t-th step

$$E(t) = E(U(t),V(t),W(t)) = \frac{1}{2}\sum_{k=1}^{N}(d_k - z_k(t))^2. \tag{13}$$

Then the time evolution of the weights of synaptic connections is determined by the gradient descent:

$$\begin{aligned} w_{ij}(t+1) &= w_{ij}(t) - \epsilon\frac{\partial E(t)}{\partial w_{ij}(t)}, \\ v_{ij}(t+1) &= v_{ij}(t) - \epsilon\frac{\partial E(t)}{\partial v_{ij}(t)}, \\ u_{ij}(t+1) &= u_{ij}(t) - \epsilon\frac{\partial E(t)}{\partial u_{ij}(t)}, \end{aligned} \tag{14}$$

which is a natural generalization of that of Section 2.1.

Before the calculation let us make a comment on the derivatives of a composite function.

Comment Let us explain with a simple example. For a composite function

$$x \to f(x) \to g(f(x)) \to E(g(f(x)))$$

the derivative is given by

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \frac{\mathrm{d}E(g(f(x)))}{\mathrm{d}g(f(x))} \frac{\mathrm{d}g(f(x))}{\mathrm{d}f(x)} \frac{\mathrm{d}f(x)}{\mathrm{d}x} = E'(g(f(x)))g'(f(x))f'(x).$$

A composite function is in general complicated, while its derivative is not so complicated.

3.2. Calculation

Let us perform the calculation of (14).

1) Case of $W(t) = (w_{ij}(t))$: From

$$\begin{aligned}\frac{\partial E(t)}{\partial w_{ij}(t)} &= \frac{\partial E(t)}{\partial z_i(t)} \frac{\partial z_i(t)}{\partial r_i(t)} \frac{\partial r_i(t)}{\partial w_{ij}(t)} \\ &= -(d_i - z_i(t))\sigma'(r_i(t))\beta_j(t) \\ &= -(d_i - z_i(t))\sigma(r_i(t))(1-\sigma(r_i(t)))\beta_j(t)\end{aligned}$$

we obtain

$$w_{ij}(t+1) = w_{ij}(t) + \epsilon(d_i - z_i(t))\sigma(r_i(t))(1-\sigma(r_i(t)))\beta_j(t).$$

2) Case of $V(t) = (v_{ij}(t))$: From

$$\begin{aligned}\frac{\partial E(t)}{\partial v_{ij}(t)} &= \sum_{k=1}^{N} \frac{\partial E(t)}{\partial z_k(t)} \frac{\partial z_k(t)}{\partial r_k(t)} \frac{\partial r_k(t)}{\partial \beta_i(t)} \frac{\partial \beta_i(t)}{\partial q_i(t)} \frac{\partial q_i(t)}{\partial v_{ij}(t)} \\ &= -\sum_{k=1}^{N} (d_k - z_k(t))\sigma'(r_k(t))w_{ki}(t)\sigma'(q_i(t))\alpha_j(t) \\ &= -\sum_{k=1}^{N} (d_k - z_k(t))\sigma(r_k(t))(1-\sigma(r_k(t))) \\ &\quad \times w_{ki}(t)\sigma(q_i(t))(1-\sigma(q_i(t)))\alpha_j(t)\end{aligned}$$

we have

$$\begin{aligned}v_{ij}(t+1) = v_{ij}(t) &+ \epsilon \sum_{k=1}^{N} (d_k - z_k(t))\sigma(r_k(t))(1-\sigma(r_k(t))) \\ &\times w_{ki}(t)\sigma(q_i(t))(1-\sigma(q_i(t)))\alpha_j(t).\end{aligned}$$

3) Case of $U(t) = (u_{ij}(t))$: From

$$\begin{aligned}\frac{\partial E(t)}{\partial u_{ij}(t)} &= \sum_{l=1}^{N}\sum_{k=1}^{M} \frac{\partial E(t)}{\partial z_l(t)} \frac{\partial z_l(t)}{\partial r_l(t)} \frac{\partial r_l(t)}{\partial \beta_k(t)} \frac{\partial \beta_k(t)}{\partial q_k(t)} \frac{\partial q_k(t)}{\partial \alpha_i(t)} \frac{\partial \alpha_i(t)}{\partial p_i(t)} \frac{\partial p_i(t)}{\partial u_{ij}(t)} \\ &= -\sum_{l=1}^{N}\sum_{k=1}^{M} (d_l - z_l(t))\sigma'(r_l(t))w_{lk}(t)\sigma'(q_k(t))v_{ki}(t)\sigma'(p_i(t))x_j\end{aligned}$$

we have

$$\begin{aligned} & u_{ij}(t+1) \\ & = u_{ij}(t) + \epsilon \sum_{l=1}^{N} \sum_{k=1}^{M} \left(d_l - z_l(t)\right) \sigma\left(r_l(t)\right) \left(1 - \sigma\left(r_l(t)\right)\right) w_{lk}(t) \\ & \times \sigma\left(q_k(t)\right) \left(1 - \sigma\left(q_k(t)\right)\right) v_{ki}(t) \sigma\left(p_i(t)\right) \left(1 - \sigma\left(p_i(t)\right)\right) x_j. \end{aligned}$$

The calculation so far is based on one input data, for simplicity. However, we must treat a lot of data simultaneously. Let us denote by *a* the number of data to be considered.

For $l = 1, 2, \cdots, a$ we set

$$\begin{aligned} \text{Input Data} \quad & \boldsymbol{x}^{(l)} = \left(x_1^{(l)}, x_2^{(l)}, \cdots, x_K^{(l)}\right)^{\mathrm{T}}, \\ \text{Teacher Signal} \quad & \boldsymbol{d}^{(l)} = \left(d_1^{(l)}, d_2^{(l)}, \cdots, d_N^{(l)}\right)^{\mathrm{T}}, \\ \text{Output Data} \quad & \boldsymbol{z}^{(l)} = \left(z_1^{(l)}, z_2^{(l)}, \cdots, z_N^{(l)}\right)^{\mathrm{T}}. \end{aligned}$$

By taking all these into consideration the square error function in (13) changes to

$$\hat{E}(t) = \sum_{l=1}^{a} E^{(l)}(t) = \sum_{l=1}^{a} \left\{ \frac{1}{2} \sum_{k=1}^{N} \left(d_k^{(l)} - z_k^{(l)}(t)\right)^2 \right\} \tag{15}$$

at the *t*-th step.

In this case the calculation is very simple. For example, since

$$w_{ij}(t+1) = w_{ij}(t) - \epsilon \frac{\partial \hat{E}(t)}{\partial w_{ij}(t)} = w_{ij}(t) - \epsilon \sum_{l=1}^{a} \frac{\partial E^{(l)}(t)}{\partial w_{ij}(t)}$$

it is only changed to

$$w_{ij}(t+1) = w_{ij}(t) + \epsilon \sum_{l=1}^{a} \left(d_i^{(l)} - z_i^{(l)}(t)\right) \sigma\left(r_i^{(l)}(t)\right) \left(1 - \sigma\left(r_i^{(l)}(t)\right)\right) \beta_j^{(l)}(t).$$

The calculation for both $v_{ij}(t)$ and $u_{ij}(t)$ is quite similar (omitted).

In the following we make a mysterious assumption.

Assumption We assume that the data in the final step of the intermediate layers are linearly independent:

$$\begin{aligned} & 1)\ a < M, \\ & 2)\ \left\{\beta^{(1)}, \beta^{(2)}, \cdots, \beta^{(a)}\right\} \text{ are linearly independent.} \end{aligned} \tag{16}$$

See **Figure 4** and the flow of data (12).

We can show that there is no critical point in our system except for $\hat{E} = 0$. Therefore, when modifying $u_{ij}(t), v_{ij}(t), w_{ij}(t)$ by the gradient descent (14) (replaced by $\hat{E}$) successively there is no danger of being trapped by points taking local minima.

Proof: The proof is essentially the same as that of Section 2.2. However, the proof is the heart of the paper, so we repeat it.

A critical point is defined by the equations

$$\begin{cases} \dfrac{\partial \hat{E}(t)}{\partial w_{ij}(t)} = 0, \\ \dfrac{\partial \hat{E}(t)}{\partial v_{ij}(t)} = 0, \\ \dfrac{\partial \hat{E}(t)}{\partial u_{ij}(t)} = 0. \end{cases}$$

From (15) we have and set

$$\begin{aligned} 0 = \frac{\partial \hat{E}(t)}{\partial w_{ij}(t)} &= -\sum_{l=1}^{a} \left(d_i^{(l)} - z_i^{(l)}(t) \right) \sigma' \left(r_i^{(l)}(t) \right) \beta_j^{(l)}(t) \\ &= \sum_{l=1}^{a} \beta_j^{(l)}(t) \left(d_i^{(l)} - z_i^{(l)}(t) \right) \sigma' \left(r_i^{(l)}(t) \right) (- \text{ is omitted}) \\ &\equiv \sum_{l=1}^{a} \beta_j^{(l)}(t) A_i^{(l)} \quad (j = 1, 2, \cdots, M) \end{aligned}$$

and it implies

$$0 = \sum_{l=1}^{a} \beta_j^{(l)}(t) A_i^{(l)} \quad (j = j_1 < j_2 < \cdots < j_a).$$

Therefore, its matrix form becomes

$$\begin{pmatrix} \beta_{j_1}^{(1)} & \beta_{j_1}^{(2)} & \cdots & \beta_{j_1}^{(a)} \\ \beta_{j_2}^{(1)} & \beta_{j_2}^{(2)} & \cdots & \beta_{j_2}^{(a)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{j_a}^{(1)} & \beta_{j_a}^{(2)} & \cdots & \beta_{j_a}^{(a)} \end{pmatrix} \begin{pmatrix} A_i^{(1)} \\ A_i^{(2)} \\ \vdots \\ A_i^{(a)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Since the determinant of coefficient matrix is non-zero from (16) we have

$$\begin{pmatrix} A_i^{(1)} \\ A_i^{(2)} \\ \vdots \\ A_i^{(a)} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

and

$$A_i^l \equiv \left(d_i^{(l)} - z_i^{(l)}(t) \right) \sigma' \left(r_i^{(l)}(t) \right) = 0 \quad (l = 1, 2, \cdots, a)$$

and $\sigma' \left(r_i^{(l)}(t) \right) \neq 0$ gives

$$d_i^{(l)} - z_i^{(l)}(t) = 0 \quad (l = 1, 2, \cdots, a).$$

Moreover, these equations are independent of $i = 1, 2, \cdots, N$, so that we finally obtain

$$\hat{E}(t) = \sum_{l=1}^{a} E^{(l)}(t) = \sum_{l=1}^{a} \left\{ \frac{1}{2} \sum_{k=1}^{N} \left(d_k^{(l)} - z_k^{(l)}(t) \right)^2 \right\} = 0.$$

Let us summarize the result :

Theorem II Under the assumption (16) there is only one global minimum $\hat{E} = 0$.

Let us emphasize that the assumption (16) is "local" because it refers to the linear independence of the data of the layer one step before the last of the flow:

$$\boldsymbol{x}=\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{pmatrix} \Rightarrow \boldsymbol{\alpha}=\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_L \end{pmatrix} \Rightarrow \boldsymbol{\beta}=\begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_M \end{pmatrix} \Rightarrow \boldsymbol{z}=\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_N \end{pmatrix}.$$

A question arises naturally.

Problem I What is the meaning of the assumption in the total flow of data ?

Last in this section let us comment on the minibatch of Deep Learning. When input data A is huge the calculation of time evolution of the weights of synaptic connections will give a heavy load to the computer. In order to alleviate it the minibatch is practical and very useful, see Figure 5.

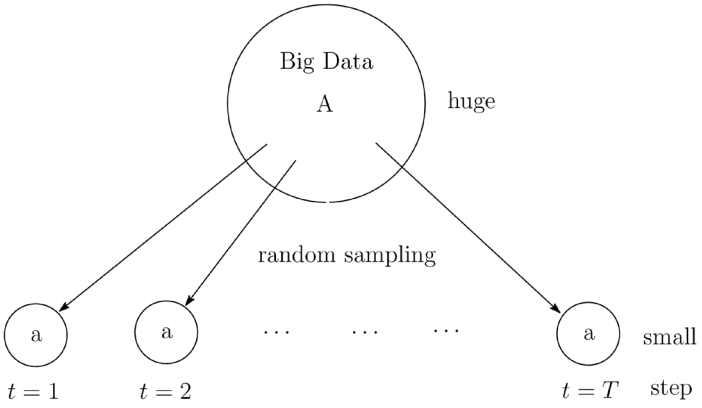

Figure 5. Minibatch.

Then a question also arises naturally.

Problem II Is there a relation between A and T?

One may conjecture

$$T=[A/a]$$

where $[k]$ is the Gauss symbol, which is the greatest integer less than or equal to k. For example, [3.14] = 3. However, I do not believe this is correct.

4. Concluding Remarks

Nowadays, studying Deep Learning is standard for university students in the world. However, it is not so easy for them because Deep Learning requires a lot of knowledge.

In this paper, we treated the minibatch of Deep Learning and gave the mathematical reinforcement to it from the viewpoint of Linear Algebra and presented some related problems. We expect that young researchers will solve the problems in the near future.

As a related topic of Artificial Intelligence, there is Information Retrieval. Since there is no more space, we only refer to some relevant works [11] [12]

[13].

Acknowledgements

We wish to thank Ryu Sasaki for useful suggestions and comments.

References

[1] Wikipedia. Deep Learning. https://en.wikipedia.org/wiki/Deep_learning

[2] Patterson, J. and Gibson, A. (2017) Deep Learning: A Practitioner's Approach. O'Reilly Media, Inc., Sebastopol, CA.

[3] Okaya, T. (2015) Deep Learning. Kodansha Ltd., Tokyo. (In Japanese)

[4] Mizoguchi, R. and Ishida, R. (2000) Artificial Intelligence. Ohmsha Ltd., Tokyo. (In Japanese)

[5] Le, Q.V., *et al.* (2012) Building High-Level Features Using Large Scale Unsupervised Learning. *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, June 27th-July 3rd, 2012, 11.

[6] Aihara, K. and Kanzaki, R. (2008) Theoretical and Engineering Approaches to Brainscience. University of Tokyo Press, Tokyo. (In Japanese)

[7] Amari, S. (2016) Brain·Heart·Artificial Intelligence. Blue Backs B-1968. Kodansha Ltd., Tokyo. (In Japanese)

[8] Fujii, K. (2014) Verhulst Model of Population Growth. Lecture Note. Yokohama City University, Yokohama. (In Japanese)

[9] Wikipedia. Gradient Descent. https://en.wikipedia.org/wiki/Gradient_descent

[10] Kasahara, K. (2000) Linear Algebra. Saiensu Ltd., Tokyo. (In Japanese)

[11] Kita, K., Tsuda, K. and Shishibori, M. (2002) Information Retrieval Algorithms. Kyoritsu Shuppann, Ltd., Tokyo. (In Japanese)

[12] Fujii, K. and Oike, H. (2015) A Superintroduction to Google Matrices for Undergraduates. *Far East Journal of Mathematical Education*, **14**, 55-68. https://doi.org/10.17654/FJMEFeb2015_055_068

[13] Fujii, K. and Oike, H. (2015) How to Understand Google from the Mathematical Point of View. Yokohama City University, Yokohama.