Weight Initialization without Local Minima in Deep Nonlinear Neural Networks

Tohru Nitta
National Institute of Advanced Industrial Science and Technology (AIST), Japan
tohru-nitta@aist.go.jp

Abstract

In this paper, we propose a new weight initialization method called even initialization for wide and deep nonlinear neural networks with the ReLU activation function. We prove that no poor local minimum exists in the initial loss landscape in the wide and deep nonlinear neural network initialized by the even initialization method that we propose. Specifically, in the initial loss landscape of such a wide and deep ReLU neural network model, the following four statements hold true: 1) the loss function is non-convex and non-concave; 2) every local minimum is a global minimum; 3) every critical point that is not a global minimum is a saddle point; and 4) bad saddle points exist. We also show that the weight values initialized by the even initialization method are contained in those initialized by both of the (often used) standard initialization and He initialization methods.

1 Introduction

Hinton et al. (2006) proposed Deep Belief Networks with a learning algorithm that trains one layer at a time. Since that report, deep neural networks have attracted attention extensively because of their human-like intelligence achieved through learning and generalization. To date, deep neural networks have produced outstanding results in the fields of image processing and speech recognition (Mohamed et al., 2009; Seide et al., 2011; Taigman et al., 2014). Moreover, their scope of application has expanded, for example, to the field of machine translation (Sutskever et al., 2014).

In using deep neural networks, finding a good initialization becomes extremely important to achieve good results. Heuristics have been used for weight initialization of neural networks for a long time. For example, a uniform distribution $U[-1/\sqrt{n}, 1/\sqrt{n}]$ has been often used where $n$ is the number of neurons in the preceding layer. Pre-training might be regarded as a kind of weight initialization methods, which could avoid local minima and plateaus (Bengio et al., 2007). However, several theoretical researches on weight initialization methods have been progressing in recent years. Glorot and Bengio (2010) derived a theoretically sound uniform distribution $U[-\sqrt{6}/\sqrt{n_i + n_{i+1}}, \sqrt{6}/\sqrt{n_i + n_{i+1}}]$ for the weight initialization of deep neural networks with an activation function which is symmetric and linear at the origin. He et al. (2015) proposed a weight initialization method (called He initialization here) with a normal distribution (either $N(0, 2/n_i)$ or $N(0, 2/(n_{i+1}))$) for the neural networks with the ReLU (Rectified Linear Unit) activation function. The above two initialization methods are driven by experiments to monitor activations and back-propagated gradients during learning.

On the other hand, local minima of deep neural networks has been investigated theoretically in recent years. Local minima cause plateaus which have a strong negative influence on learning in deep neural networks. Dauphin et al. (2014) experimentally investigated the distribution of the critical points of a single-layer MLP and demonstrated that the possibility of existence of local minima with large error (i.e., bad or poor local minima) is very small. Choromanska et al. provided a theoretical justification for the work of Dauphin et al. (2014) on a deep neural network with ReLU units using
the spherical spin-glass model under seven assumptions (Choromanska, Henaff, Mathieu, Arous & LeCun, 2015). Choromanska et al. also suggested that discarding the seven unrealistic assumptions remains an important open problem (Choromanska, LeCun & Arous, 2015). Kawaguchi (2016) discarded most of these assumptions and proved that the following four statements for a deep nonlinear neural network with only two out of the seven assumptions: 1) the loss function is non-convex and non-concave, 2) every local minimum is a global minimum, 3) every critical point that is not a global minimum is a saddle point, and 4) bad saddle points exist.

In this paper, we propose a new weight initialization method (called even initialization) for wide and deep nonlinear neural networks with the ReLU activation function: weights are initialized independently and identically according to a probability distribution whose probability density function is even on $[-1/n, 1/n]$ where $n$ is the number of neurons in the layer. Using the research results presented by Kawaguchi (2016), we prove that no poor local minimum exists in the initial loss landscape in the wide and deep nonlinear neural network initialized by the even initialization method that we propose. We also show that the weight values initialized by the even initialization method are contained in those initialized by both of the (often used) standard initialization and He initialization methods.

## 2 Weight Initialization without Local Minima

In this section, we propose a weight initialization method for a wide and deep neural network model. There is no local minimum in the initial weight space of the wide and deep neural network initialized by the weight initialization method. The weight values initialized by the weight initialization method are contained in those initialized by both of the standard initialization and He initialization methods. In other words, there exists an interval of initial weights where there is no local minimum in both cases of the standard initialization and He initialization methods.

### 2.1 Kawaguchi Model

This subsection presents a description of the deep nonlinear neural network model analyzed by Kawaguchi (2016) (we call it Kawaguchi model here). We will propose a new weight initialization method for the Kawaguchi model in the subsection 2.2.

First, we consider the following neuron. The net input $U_n$ to a neuron $n$ is defined as: $U_n = \sum_m W_{nm} I_m$, where $W_{nm}$ represents the weight connecting the neurons $n$ and $m$, $I_m$ represents the input signal from the neuron $m$. It is noteworthy that biases are omitted for the sake of simplicity.

The output signal is defined as $\phi(U_n)$ where $\phi(u) \overset{\text{def}}{=} \max(0,u)$ for any $u \in \mathbb{R}$ and is called Rectified Linear Unit (ReLU, $\mathbb{R}$ denotes the set of real numbers).

The deep nonlinear neural network described in (Kawaguchi, 2016) consists of such neurons described above (Fig. 1). The network has $H + 2$ layers ($H$ is the number of hidden layers). The activation function $\psi$ of the neuron in the output layer is linear, i.e., $\psi(u) = u$ for any $u \in \mathbb{R}$. For any $0 \leq k \leq H + 1$, let $d_k$ denote the number of neurons of the $k$-th layer, that is, the width of the...
The $p$-th path

Figure 2: Image of a path in the deep neural network model (Kawaguchi, 2016): $X_i$ denotes the $i$-th input training pattern, and $\hat{Y}_i(\Theta, X_i)_j$ stands for the actual output of output neuron $j$.

$k$-th layer where the 0-th layer is the input layer and the $(H + 1)$-th layer is the output layer. Let $d_x = d_0$ and $d_y = d_{H+1}$ for simplicity.

Let $(X, Y)$ be the training data where $X \in \mathbb{R}^{d_x \times m}$ and $Y \in \mathbb{R}^{d_y \times m}$ and where $m$ denotes the number of training patterns. We can rewrite the training data as $\{(X_i, Y_i)\}_{i=1}^m$ where $X_i \in \mathbb{R}^{d_x}$ is the $i$-th input training pattern and $Y_i \in \mathbb{R}^{d_y}$ is the $i$-th output training pattern. Let $W_k$ denote the weight matrix between the $(k-1)$-th layer and the $k$-th layer for any $1 \leq k \leq H + 1$. Let $\Theta$ denote the one-dimensional vector which consists of all the weight parameters of the deep nonlinear neural network.

Kawaguchi specifically examined a path from an input neuron to an output neuron of the deep nonlinear neural network (Fig. 3), and expressed the actual output of output neuron $j$ of the output layer of the deep nonlinear neural network for the $i$-th input training pattern $X_i \in \mathbb{R}^{d_x}$ as

$$\hat{Y}_i(\Theta, X_i)_j = q \sum_{p=1}^{\Psi} [X_i]_{(j,p)} [Z_i]_{(j,p)} \prod_{k=1}^{H+1} w_{(j,p)}^{(k)} \in \mathbb{R}$$

where $\Psi$ represents the total number of paths from the input layer to output neuron $j$, $[X_i]_{(j,p)} \in \mathbb{R}$ denotes the component of the $i$-th input training pattern $X_i \in \mathbb{R}^{d_x}$ that is used in the $p$-th path to the $j$-th output neuron. Also, $[Z_i]_{(j,p)} \in \{0, 1\}$ represents whether the $p$-th path to the output neuron $j$ is active or not for each training pattern $i$ as a result of ReLU activation. $[Z_i]_{(j,p)} = 1$ means that the path is active, and $[Z_i]_{(j,p)} = 0$ means that the path is inactive. $w_{(j,p)}^{(k)} \in \mathbb{R}$ is the component of the weight matrix $W_k \in \mathbb{R}^{d_x \times d_{k-1}}$ that is used in the $p$-th path to the output neuron $j$.

The objective of the training is to find the parameters which minimize the error function defined as

$$L(\Theta) = \frac{1}{2} \sum_{i=1}^{m} E_Z \| \hat{Y}_i(\Theta, X_i) - Y_i \|^2$$

where $\| \cdot \|$ is the Euclidean norm, that is, $\|u\| = \sqrt{u_1^2 + \cdots + u_N^2}$ for a vector $u = (u_1 \cdots u_N)^T \in \mathbb{R}^N$, and $\hat{Y}_i(\Theta, X_i) \in \mathbb{R}^{d_y}$ is the actual output of the output layer of the deep nonlinear neural network for the $i$-th training pattern $X_i$. The expectation in Eq. (2) is made with respect to random vector $Z = \{[Z_i]_{(j,p)}\}$.

The Kawaguchi model has been analyzed based on the two assumptions.

**A1p-m** $P([Z_i]_{(j,p)} = 1) = \rho$ for all $i$ and $(j, p)$ where $\rho \in \mathbb{R}$ is a constant. That is, $[Z_i]_{(j,p)}$ is a Bernoulli random variable.

**A5u-m** $Z$ is independent of the input $X$ and the parameter $\Theta$.

A1p-m and A5u-m are weaker ones of the two assumptions A1p and A5u in (Choromanska, Henaff, Mathieu, Arous & LeCun, 2015), respectively. Assumption A5u-m is used for the proof of Corollary 3.2 presented by Kawaguchi (2016). Strictly speaking, the following assumption A5u-m-1 suffices for the proof instead of the assumption A5u-m described above.
\textbf{A5u-m-1} For any \( i \) and any \((j, p)\), \([Z_{i(j,p)}] \in \{0, 1\}\) is independent of the \(i\)-th input training pattern \(X_i \in \mathbb{R}^{d_x}\) and the sequence of the weights on the \(p\)-th path \(\{w_{i(j,p)}^{(k)} \in \mathbb{R}\}_{k=1}^{H+1}\), where \(w_{i(j,p)}^{(k)}\) is the weight between the layer \(k - 1\) and the layer \(k\) on the \(p\)-th path \((k = 1, \ldots, H + 1)\).

Actually, according to assumption A5u-m-1, we designate it as

\[
\begin{align*}
    E_Z \left[ Y_i(\Theta, X_{i}) \right] &= E_Z \left[ \sum_{Z_{i(j,p)}} [X_{i}]_{(j,p)} [Z_{i}]_{(j,p)} \prod_{k=1}^{H+1} w_{i(j,p)}^{(k)} \right] \quad \text{(from Eq. (1))} \\
    &= q \sum_{p=1}^{\psi} [X_{i}]_{(j,p)} E_Z \left[ [Z_{i}]_{(j,p)} \right] \prod_{k=1}^{H+1} w_{i(j,p)}^{(k)} \quad \text{(from the assumption A5u-m-1) (3)}
\end{align*}
\]

\section{2.2 Even Initialization}

This subsection proposes a new weight initialization method for the Kawaguchi model described in the subsection 2.1. We assume that the width of the deep nonlinear neural network is sufficiently large, that is, \(d_0 = d_x, d_1, \ldots, d_{H-1}\) are sufficiently large (Fig. 1), and that each element of the \(i\)-th input training pattern \(X_i\) takes a value between \(-\alpha\) and \(\alpha\), that is, \(X_i \in \mathbb{I}^{d_x}\) for any \(i\) where \(I = [-\alpha, \alpha]\) and \(\alpha\) is a positive real number. Then we propose the following weight initialization scheme.

**Even Initialization** Elements \(\{w_n\}_{n=1}^{d_x-1}\) of the weight vector \(w = (w_1 \cdots w_{d_x-1})^T\) of any hidden neuron in the \(k\)-th hidden layer are initialized independently and identically according to a probability distribution whose probability density function \(f_k : J_k \rightarrow [0, +\infty)\) is an even function where \(J_k = [-1/d_{k-1}, 1/d_{k-1}]\), that is, \(f_k(-x) = f_k(x)\) for any \(x \in J_k (k = 1, \ldots, H)\).

We designate it as \textit{even initialization}.

\textbf{Remarks} (a) The expectation of each weight is zero because the probability density function is an even function. (b) The probability distribution in the even initialization can be a normal distribution with zero mean or a uniform distribution. (c) For the weight vectors \(w_1 = (w_{1,1} \cdots w_{1,M})^T, w_2 = (w_{2,1} \cdots w_{2,N})^T\) of any two hidden neurons, the probability distribution which \(w_{1k}\) obeys can be different from that which \(w_{2l}\) obeys \((k = 1, \ldots, M, l = 1, \ldots, N)\) (Fig. 3).

\section{2.3 Analysis}

This subsection presents an analysis of the initial loss landscape of the Kawaguchi model initialized by the even initialization method described in Section 2.2.

We prove in the following that the two assumptions A1p-m and A5u-m (A5u-m-1) are satisfied in the Kawaguchi model initialized by the even initialization.

**Theorem 1** For any training pattern \(i\), any output neuron \(j\), and any path \(p\) from an input neuron to the output neuron \(j\) in the Kawaguchi model initialized by the even initialization method,

\[
P ([Z_{i}]_{(j,p)} = 1) = \frac{1}{2^H} \quad (4)
\]

![Figure 3: Probability distribution which weight \(w_{1k}\) obeys can be different from the one which \(w_{2l}\) obeys.](image)
The $p$-th path

Figure 4: A deep nonlinear neural network with $H$ hidden layers where $w_{jk}$ is the weight vector of hidden neuron $j_k$, hidden neurons $j_1, \cdots, j_H$ are on the $p$-th path, $d_k$ is the number of neurons of the $k$-th layer, $d_x = d_0$ and $d_y = d_{H+1}$.

where $H$ is the number of hidden layers ($H \geq 1$).

Proof. Denote by $j_1, \cdots, j_H$ the hidden neurons on path $p$ where $j_k$ is the hidden neuron in the $k$-th hidden layer ($k = 1, \cdots, H$) (Fig. 4). Then,

\[
P\left(\left[Z_i^{(j,p)} = 1\right]\right) = P\left(\text{Net input to the hidden neuron } j_k > 0 \ (k = 1, \cdots, H)\right)
= P\left(\begin{array}{c}
X_i^T w_{j_1} > 0, \\
\varphi(U_1^{(1)}) \cdots \varphi(U_{d_1}^{(1)}) w_{j_2} > 0, \\
\cdots, \\
\varphi(U_{d_1}^{(H-1)}) \cdots \varphi(U_{d_{H-1}}^{(H-1)}) w_{j_H} > 0
\end{array}\right)
\]

where $X_i \in I^{d_x}$ is the $i$-th input training pattern, $w_{jk} \in J^{d_k-1}$ is the weight vector of the hidden neuron $j_k$ in the hidden layer $k$, and $U_l^{(k)}$ is the net input to the hidden neuron $l$ in the hidden layer $k$ (Fig. 5).

We prove by mathematical induction that Eq. 4 holds true.

[For $H = 1$] This case corresponds to a three-layered neural network. It follows that

\[
P\left(\left[Z_i^{(j,p)} = 1\right]\right) = P(\begin{array}{c}
X_i^T w_{j_1} > 0
\end{array}) \quad \text{(from Eq. 5)}
= \frac{1}{2}
\]

Figure 5: Relationship between the weight vector $w_{jk}$ and the net inputs $U_l^{(k-1)}$, $\cdots, U_{d_{k-1}}^{(k-1)}$. 

\[
\text{Net input } U_{d_{k-1}}^{(k-1)} \quad \text{Net input } U_l^{(k-1)} \quad \text{Net input } U_{d_k}^{(k-1)} 
\]
where \( j_1 \) is a hidden neuron on path \( p \). We can see below that the last equality of Eq. (6) holds true. For a given input training pattern \( X_i \in I^{d_x} \), \( \{ w_{j_1} \in J_1^{d_x} | X_i^T w_{j_1} > 0 \} \) is a half-open hyperspace with a normal vector \( X_i \) through the origin in a \( d_x \)-dimensional Euclidean space (Fig. 6). According to the even initialization scheme, the random variables \( w_{j_1}, \ldots, w_{j_1 d_x} \) which are the components of the weight vector \( w_{j_1} = (w_{j_1 1} \cdots w_{j_1 d_x})^T \) of the hidden neuron \( j_1 \) are i.i.d., and the probability density function \( f_{j_1} : J_1 \to [0, +\infty) \) of the hidden neuron \( j_1 \) is an even function (\( w \)). Consequently, \( P \left( X_i^T w_{j_1} > 0 \right) = P \left( X_i^T w_{j_1} \leq 0 \right) \), which means \( P \left( X_i^T w_{j_1} > 0 \right) = 1/2 \).

[For \( H \)] This case corresponding to a deep nonlinear neural network with \( H \) hidden layers, we show that if the case of \( H - 1 \) holds, then case \( H \) also holds. Assuming that the case of \( H - 1 \) holds, then

\[
P \left( [Z_{i}]_{(j,p)} = 1 \right) = P \left( X_i^T w_{j_1} > 0, \left[ \varphi(U_{i}^{(1)}) \cdots \varphi(U_{i}^{(H-1)}) \right] w_{j_2} > 0, \ldots, \left[ \varphi(U_{i}^{(1)}) \cdots \varphi(U_{i}^{(H-1)}) \right] w_{j_H} > 0 \right) \quad \text{from Eq. (6)}
\]

\[
= P \left( X_i^T w_{j_1} > 0, \left[ \varphi(U_{i}^{(1)}) \cdots \varphi(U_{i}^{(H-1)}) \right] w_{j_2} > 0, \cdots, \left[ \varphi(U_{i}^{(1)}) \cdots \varphi(U_{i}^{(H-1)}) \right] w_{j_H} > 0 \right) \cdot P \left( X_i^T w_{j_1} > 0 \right) + P \left( X_i^T w_{j_1} > 0, \left[ \varphi(U_{i}^{(1)}) \cdots \varphi(U_{i}^{(H-1)}) \right] w_{j_2} > 0, \cdots, \left[ \varphi(U_{i}^{(1)}) \cdots \varphi(U_{i}^{(H-1)}) \right] w_{j_H} > 0 \right) \cdot P \left( X_i^T w_{j_1} \leq 0 \right)
\]

\[
= P \left( \left[ \varphi(U_{i}^{(1)}) \cdots \varphi(U_{i}^{(H-1)}) \right] w_{j_2} > 0, \cdots, \left[ \varphi(U_{i}^{(1)}) \cdots \varphi(U_{i}^{(H-1)}) \right] w_{j_H} > 0 \right) \cdot P \left( X_i^T w_{j_1} > 0 \right) \cdot P \left( X_i^T w_{j_1} > 0 \right) \cdot P \left( X_i^T w_{j_1} > 0 \right).
\]

Here, the first term of the right-hand-side of Eq. (7) represents the probability that the path passing through the \( H - 1 \) hidden neurons \( j_2, \ldots, j_H \) for the input training pattern \( [\varphi(U_{i}^{(1)}) \cdots \varphi(U_{i}^{(H-1)})]^T \in R^{d_H} \) such that \( U_{j_1}^{(1)} = X_i^T w_{j_1} > 0 \) is active. Also, for any \( 1 \leq s \leq d_1 \), by denoting \( X_i = (x_1 \cdots x_{d_0})^T \) and \( w_s = (w_1 \cdots w_{d_0}) \) for the sake of simplicity, \( |\varphi(U_{s}^{(1)})| \leq |U_{s}^{(1)}| = |X_i^T w_s| \leq \sum_{i=1}^{d_0} |x_i||w_s| \leq \alpha \sum_{i=1}^{d_0} (1/d_0) = \alpha \). Hence, according to the assumption of mathematical induction, the first term of the right-hand-side of Eq. (7) is equal to \((1/2)^{H-1}\). In addition,
We prove below by mathematical induction that the approximate equality
\[ (10) \]
which means that the case of \( j \)
\[ (1) \]
where \( j \)
\[ p \]
weights on path \( p \)
\[ \text{Proof} \]
the probability that path \( p \)
\[ \text{Figure 7: Relationship between the weight vector } w_{jk+1} \text{ and the net inputs } U_{1}^{(k)}, \ldots, U_{d_{k}}^{(k)} \text{ in Eq. (10)}. \]
the second term of the right-hand-side of Eq. (7) is equal to 1/2 from Eq.(6). Therefore,
\[ \frac{1}{2^{H+1}}, \]
which means that the case of \( H \) indeed holds. Therefore, by mathematical induction, Eq. (4) holds for any \( H \geq 1 \).

Theorem 1 states that assumption A1p-m holds. Because
\[ \lim_{H \to +\infty} P\left( [Z_{i}]_{(j,p)} = 1 \right) = \lim_{H \to +\infty} \frac{1}{2^{H+1}} = 0, \]
the probability that path \( p \) is active decreases exponentially. It converges to zero as the number of hidden layers \( H \) increases. The probability that path \( p \) is active decreases by half when a hidden layer is added.

**Theorem 2** For any training pattern \( i \), any output neuron \( j \), and any path \( p \) from an input neuron to the output neuron \( j \) in the Kawaguchi model initialized by the even initialization method, random variable \( [Z_{i}]_{(j,p)} \) is independent of the sequence of weights on path \( p \).

**Proof.** Take training pattern \( i \), output neuron \( j \), and path \( p \) from an input neuron to output neuron \( j \) of the Kawaguchi model arbitrarily and fix them. Denote by \( j_{0}, \ldots, j_{H} \) the neurons on path \( p \) where \( j_{k} \) is the neuron in the \( k \)-th layer \( (k = 0, \ldots, H) \). Let also \( w_{j_{0}}, \ldots, w_{j_{H+1}} \) denote \( H + 1 \) weights on path \( p \) where \( w_{j_{H+1}} \) is the weight between the layer \( k \) and the layer \( k + 1 \) on path \( p \) \( (k = 0, \ldots, H) \) (Fig. 7). Then, for any \( \lambda_{1}, \ldots, \lambda_{H+1} \in \mathbb{R} \),
\[ P\left( [Z_{i}]_{(j,p)} = 1 \left| w_{j_{1}j_{0}} = \lambda_{1}, \ldots, w_{j_{H+1}j_{H}} = \lambda_{H+1} \right. \right) \]
\[ = P\left( X_{i}^{T} w_{j_{1}} > 0, \left[ \varphi(U_{1}^{(1)}) \cdots \varphi(U_{d_{1}}^{(1)}) \right] w_{j_{2}} > 0, \ldots, \left[ \varphi(U_{1}^{(H-1)}) \cdots \varphi(U_{d_{H-1}}^{(H-1)}) \right] w_{j_{H+1}} > 0 \right| w_{j_{1}j_{0}} = \lambda_{1}, \ldots, w_{j_{H}j_{H-1}} = \lambda_{H} \right) \text{ (from Eq. (5))} \]
\[ \overset{\text{w}_{j_{H+1}j_{H}} = \lambda_{H+1} \text{ is removed because it is independent of } [Z_{i}]_{(j,p)}}{=} \frac{1}{2^{H}}. \]
We prove below by mathematical induction that the approximate equality \( (=) \) in Eq. (10) holds true.
[For \( H = 1 \) This case corresponds to a three-layered neural network. For the sake of simplicity, we let \( X_{i} = (x_{1}, \ldots, x_{j_{0}}, \ldots, x_{d_{0}})^{T} \) and \( w_{j_{1}} = (w_{1}, \ldots, w_{j_{0}}, \ldots, w_{d_{0}})^{T} \) where \( w_{j_{0}} = w_{j_{1}j_{0}} \). Then,\]
\[ P\left( X_{i}^{T} w_{j_{1}} > 0 \left| w_{j_{1}j_{0}} = \lambda_{1} \right. \right) \]
\[
\begin{align*}
\mathbb{P}
&= P\left(x_1w_1 + \cdots + x_jw_{j,0} + \cdots + x_{d_0}w_{d_0} > 0 \mid w_{j,0} = \lambda_1\right) \\
&= P\left(x_1w_1 + \cdots + x_jw_{j,0} + \cdots + x_{d_0}w_{d_0} > 0 \mid w_{j,0} = \lambda_1\right) \\
&= P\left(x_1w_1 + \cdots + x_jw_{j,0} + \cdots + x_{d_0}w_{d_0} > 0 \right) \\
&\quad (w_{j,0} = \lambda_1 \text{ is removed because it is independent of the other weights}) \\
&= P\left(x_1w_1 + \cdots + x_{j-1}w_{j-1} + x_{j+1}w_{j+1} + \cdots + x_{d_0}w_{d_0} > -x_j\lambda_1\right) \\
&\geq P\left(x_1w_1 + \cdots + x_{j-1}w_{j-1} + x_{j+1}w_{j+1} + \cdots + x_{d_0}w_{d_0} > 0 \right) \\
&\quad (\text{because } |x_j\lambda_1| = |x_j\lambda_1| \leq \alpha \cdot (1/d_0) \text{ and the number of input neurons } d_0 \text{ is sufficiently large}) \\
&= \frac{1}{2} \quad \text{(for the same reason that the last equality of Eq. (6) holds true.)} \\
\end{align*}
\]

[For \( H \)]

This case corresponding to a deep nonlinear neural network with \( H \) hidden layers, we show that if the case of \( H - 1 \) holds, then case \( H \) also holds. Assuming that the case of \( H - 1 \) holds, then

\[
\begin{align*}
\mathbb{P}
&= P\left(X_j^T w_j > 0, \left[\varphi(U^{(1)}_1) \cdots \varphi(U^{(1)}_{d_1})\right] w_{j_2} > 0, \cdots, \left[\varphi(U^{(H-1)}_1) \cdots \varphi(U^{(H-1)}_{d_{H-1}})\right] w_{j_H} > 0 \right) \\
&= P\left(X_j^T w_j > 0 \mid w_{j_1,0} = \lambda_1, \cdots, w_{j_H,0} = \lambda_H\right) \cdot \mathbb{P}\left(X_j^T w_j > 0 \mid w_{j_1,0} = \lambda_1\right). \\
&= P\left(X_j^T w_j > 0 \mid w_{j_1,0} = \lambda_1, \cdots, w_{j_H,0} = \lambda_H\right) \cdot \mathbb{P}\left(X_j^T w_j > 0 \right) \\
&\quad (w_{j_2,1} = \lambda_2, \cdots, w_{j_H,0} = \lambda_H \text{ are removed because they are independent of } w_{j_1}) \\
\end{align*}
\]

Here, the first term of the right-hand-side of Eq. (12) represents the probability that the path passing through the \( H - 1 \) hidden neurons \( j_2, \cdots, j_H \) for the input training pattern \([\varphi(U^{(1)}_1) \cdots \varphi(U^{(1)}_{d_1})]T \in R^{d_1}\) such that \( U^{(1)}_{j_1} = X_j^T w_j > 0 \) and \( w_{j_1,0} = \lambda_1 \) is active. Also, for any \( 1 \leq s \leq d_1 \), by denoting \( X_s = (x_1 \cdots x_{d_0})^T \) and \( w_s = (w_1 \cdots w_{d_0}) \) for the sake of simplicity, \(|\varphi(U^{(1)}_s)| \leq |U^{(1)}_s| = |X_s^T w_s| \leq \sum_{d=1}^{d_0} |x_d||w_s| \leq \alpha \sum_{d=1}^{d_0} (1/d_0) = \alpha \). Hence, according to the assumption of mathematical induction, the first term of the right-hand-side of Eq. (12) is nearly equal to \((1/2)^{H-1}\). In addition, the second term of the right-hand-side of Eq. (12) is nearly equal to 1/2 from Eq. (11). So,

\[
\begin{align*}
\mathbb{P}
&= P\left(X_j^T w_j > 0, \left[\varphi(U^{(1)}_1) \cdots \varphi(U^{(1)}_{d_1})\right] w_{j_2} > 0, \cdots, \left[\varphi(U^{(H-1)}_1) \cdots \varphi(U^{(H-1)}_{d_{H-1}})\right] w_{j_H} > 0 \right) \\
&= \left(\frac{1}{2}\right)^{H-1} \cdot \frac{1}{2} \\
&= \frac{1}{2^H}. \\
\end{align*}
\]

which means that the case of \( H \) indeed holds. Thus, by mathematical induction, Eq. (10) holds for any \( H \geq 1 \). Therefore,

\[
\begin{align*}
\mathbb{P}
&= P\left([Z_{i}]_{(j,p)} = 1 \mid w_{j_1,0} = \lambda_1, \cdots, w_{j_H,0} = \lambda_H\right) \\
&= \frac{1}{2^H} \quad \text{(from Eq. (10))} \\
&= P\left([Z_{i}]_{(j,p)} = 1 \right). \\
\end{align*}
\]
This completes the proof.

**Theorem 3** For any training pattern $i$, any output neuron $j$, and any path $p$ from an input neuron to the output neuron $j$ in the Kawaguchi model initialized by the even initialization method, random variable $[Z_i]_{(j,p)}$ is independent of the input training signal $X_i$.

**Proof.** Take training pattern $i$, output neuron $j$, path $p$ from an input neuron to output neuron $j$ of the deep nonlinear neural network in the Kawaguchi model initialized by the even initialization method, and $\mu \in \mathbb{R}^{d_r}$ arbitrarily and fix them. Then, in the same mode of the proof of Theorem 1 it is apparent that

$$P ([Z_i]_{(j,p)} = 1 \mid X_i = \mu) = \frac{1}{2^H}. \quad (15)$$

Therefore, it follows that

$$P ([Z_i]_{(j,p)} = 1) = \frac{1}{2^H} \quad \text{(from Theorem 1)}$$

$$= P ([Z_i]_{(j,p)} = 1 \mid X_i = \mu) \quad \text{(from Eq. (15))} \quad (16)$$

holds true. Eq. (16) completes the proof.

Theorem 2 and Theorem 3 state that assumption A5u-m-1 holds in the initial loss landscape of the Kawaguchi model initialized by the even initialization method.

Thus, it follows from Theorem 1 that assumptions A1p-m and A5u-m (A5u-m-1) hold in the initial loss landscape of the Kawaguchi model initialized by the even initialization method.

3 Discussion and Related Work

In this section, we compare the even initialization proposed in the subsection 2.2 with the existing methods. First, let us briefly review several existing methods on the initialization for weights in neural networks.

The following uniform distribution has been often used for setting the initial value for a weight of a neuron:

$$U \left[ \frac{-1}{\sqrt{n}}, \frac{1}{\sqrt{n}} \right] \quad (17)$$

where $n$ is the number of neurons in the preceding layer (we call it standard initialization here). To the best of the authors’ knowledge, it is a heuristics.

In contrast, the following two weight initialization methods have been derived theoretically. Glorot and Bengio (2010) proposed the normalized initialization: the initial value of each weight between the layer $i$ and the layer $i + 1$ is set according to the uniform distribution

$$U \left[ \frac{-\sqrt{6}}{\sqrt{n_i + n_{i+1}}}, \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}} \right] \quad (18)$$

independently where $n_i$ is the number of neurons in the layer $i$. Eq. (18) was derived by keeping the variance of the output vector in each layer equal and keeping the variance of the back-propagated gradients equal, respectively, for the purpose of avoiding their saturation under the assumption that the variances of the input signals are all the same. The object of the normalized initialization is neural networks with symmetric activation function $f$ such that $\partial f(x)/\partial x|_{x=0} = 1$ (for example, sigmoid function and tanh function). Thus, the ReLU activation function is out of scope.

He et al. (2015) proposed a weight initialization method for neural networks with the ReLU activation function: the initial value of each weight between the layer $i$ and the layer $i + 1$ is set according to either of the normal distributions

$$N \left( 0, \frac{2}{n_i} \right) \text{ or } N \left( 0, \frac{2}{n_{i+1}} \right) \quad (19)$$
Figure 8: Schematic relationship among the standard initialization, He initialization and the even initialization. 99.73% of the weight values initialized by He initialization lie within the interval $[-3\sqrt{2}/\sqrt{n}, 3\sqrt{2}/\sqrt{n}]$ which is three standard deviations of the mean.

independently where $n_i$ is the number of neurons in the layer $i$. Eq. (19) was derived by keeping the variance of the net input vector in each layer equal and keeping the variance of the back-propagated gradients equal, respectively, for the purpose of avoiding their saturation.

The normalized initialization is inappropriate for a comparison object of the even initialization because it is invalid for neural networks with the ReLU activation function as described above. Then we compare below the even initialization method with the standard initialization and He initialization. A schematic relationship among the standard initialization, He initialization and the even initialization is shown in Fig. 8. The interval of the initial values of weights initialized by the even initialization is contained in both of those of the standard initialization and He initialization because $1/n \leq 1/\sqrt{n}$ holds true for any $n = 1, 2, \ldots$ where $n$ is the number of neurons in the preceding layer, and the probability distribution which weights obey in the even initialization can be a normal distribution with zero mean or a uniform distribution. Conversely, there exists an interval which does not have any local minimum in both of the intervals of initial weight values of the standard initialization and He initialization. As a numerical example, $[-3\sqrt{2}/\sqrt{n}, 3\sqrt{2}/\sqrt{n}] \approx [-0.424, 0.424]$ (He initialization), $[-1/\sqrt{n}, 1/\sqrt{n}] = [-0.1, 0.1]$ (standard initialization) and $[-1/n, 1/n] = [-0.01, 0.01]$ (even initialization) when $n = 100$: the number of neurons in the preceding layer is 100.

4 Conclusions

We introduced a weight initialization scheme called even initialization for the wide and deep ReLU neural network model, and proved using the research results presented by Kawaguchi (2016) that no poor local minimum exists in the initial loss landscape in the wide and deep nonlinear neural network initialized by the even initialization method. We also elucidated that the weight values initialized by the even initialization method are contained in those initialized by both of the standard initialization and He initialization methods. We clarified the essential property of the even initialization theoretically. Applying the even initialization to large-scale real-world problems is a future topic.
References

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19 (NIPS’06)*, (B. Schölkopf, J. Platt, and T. Hoffman, eds.), 153-160, 2007.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. The loss surfaces of multilayer networks. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 192-204, 2015.

Choromanska, A., LeCun, Y., and Arous, G. B. Open problem: the landscape of the loss surfaces of multilayer networks. In *Proceedings of The 28th Conference on Learning Theory*, 1756-1760, 2015.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, 2933-2941, 2014.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, 249-256, 2010.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 1026-1034, 2015.

Hinton, G. E., Osindero, S., and Teh, Y. A fast learning algorithm for deep belief nets. *Neural Computation*, 18: 1527-1554, 2006.

Kawaguchi, K. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems 29*, 2016.

Mohamed, A-R, Dahl, G. E., and Hinton, G. E. Deep belief network for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

Seide, F., Li, G., and Yu, D. Conversational speech transcription using context-dependent deep neural networks. In *Proc. Interspeech*, 437-440, 2011.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, 3104-3112, 2014.

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In *Proc. Conference on Computer Vision and Pattern Recognition*, 1701-1708, 2014.