# Convergence Analysis on Neural Network

Tianyi Chen, tchen59@jhu.edu

May 10, 2016

**Abstract**

In this project, I analyze a serie of training schemes for neural network. Based on analyzations, two widely used constant learning rate training schemes can not guarantee any form of convergence. A modified training scheme with backtracking is introduced. Convergence to stationary point of this training scheme with back tracking is derived in this report. A further modified backtracking training scheme with noised gradient is proved to guarantee convergence to stationary point or converge in "liminf" form with some assumptions. Another decaying learning rate training scheme with noised gradient is shown to converge to a stationary point almost surely. Finally, two Recurrent Neural Networks(LSTM, GRU) on language model is discussed briefly not in detail, since it has been used as a course project for another course, and is not very relative to topic of convergence analysis.

## 1 Berief Description of Neural Network

In 1943, Warren McCulloch firstly proposed a computational model for neural networks based on threshold logic [1]. Frank Rosenblatt created perceptron, a two-layer computer learning network using simple addition and subtraction [2]. Due to limited computational resource, neural network research slowed until computers achieved greater power to effectively handle the required long run time. In 1975, Werbos proposed a key algorithm called backpropagation to generally train multi-layer neural network quickly [3].

So far, deep feedforward neural network, recurrent neural network, deep belief network become the most popular neural network types. Deep feedforward neural network, including MLP(Multiple Layer Perceptron) can handle a series of classification problem as SVM[4]. Unlike the independence among input(output) units in deep feedforward network, Recurrent Neural Network has a “memory” to capture information about previous conditions, which is widely used in tasks of NLP. Gated recurrent neural network, like LSTM, GRU, are proposed to solve the issue in standard RNN, like the failure of learning in long time lags [6]. Deep belief network is a probabilistic network, stacked by Restricted Boltzmann Machine[5], often used as pretrain tool for MLP.

In recent years, due to the contribution of NVIDIA, e.g. donate Devbox to many companies, and current public cloud platforms, like AWS of Amazon, GPU accelerated Neural Network implementation mainly on CUDA is developing fast. GPU can accelerate training speed dramatically on scalable problems by iterating through the whole data set during backpropagation in parallel.

## 2 Constant Learning Rate Training Schemes

To analyze the convergence of neural network, it is necessary to understand the training procedures in neural network. So far, the training procedures I met can be catagorized into two generic schemes: **Training Scheme 1**, **Training Scheme 2**. **Training Scheme 1** updates weight variables after back propagation through the whole network. It is used for training Deep Feedforward Network, and Recurrent Neural Network(backpropagation through time). **Training Scheme 2** updates weight variables for each layer separably, used for training Deep Belief Network.

**Algorithm 1** Training Scheme 1

```
1: procedure INPUT DATASET $X$
2:     Initialize weights, bias parameter.
3: loop for $k = 1, 2, 3, \cdots$:
4:     For each $x \in X$:
5:         Forward propagate to get predicted output $\hat{y}$
6:         Calculate loss error
7:         Back propagate(through time) to calculate gradients for each variable
8:     Update each variable $Var(k)$ by:

           $$Var(k+1) = Var(k) - lr \cdot Grad(Var(k))$$

9:     if $k >=$ maximum training epoches then
10:        Stop;
11:    else
12:        goto Line 4.
```

**Algorithm 2** Training Scheme 2

```
1: procedure INPUT DATASET $X$
2:     Initialize weights, bias parameter.
3: loop for $k = 1, 2, 3, \cdots$:
4:     For each layer:
5:         Forward propagate to get predicted output $\hat{y}$ by input of current layer
6:         Calculate loss error
7:         Back propagate to calculate gradients for each variable in current layer
8:         Update each variable $Var(k)$ in current layer by:
               $$Var(k+1) = Var(k) - lr \cdot Grad(Var(k))$$
9:         Use the output on updated current layer as the input of next layer
10:    if $k >=$ maximum training epoches then
11:        Stop;
12:    else
13:        goto Line 4.
```

## 3 Convergence Analysis

In this section, I will analyze the two training schemes shown in Section 2, and build several new training schemes.

### 3.1 Notations

The following notations will occur in the rest sections:

1. $D$: Number of layers in neural network, including input layer, hidden layers, output layer.
2. $W_i$: Weight matrix for the edges between $ith$ layer, and $i+1th$ layer, $i = 1, 2, \cdots, D-1$
3. $M$: The number of samples of training dataset.
4. $N$: The number of features of input data.
5. $x_j$: The $jth$ sample of training dataset $X$.
6. $y_j$: The output of $x_j$ on neural network
7. $b_i$: Bias Neuron for $ith$ layer. (It is not noise.)
8. $L$: Loss function
9. $\sigma(x)$: sigmoid function value of vector $x$ elementwise;
10. $softmax(x)$: softmax function value of vector $x$ elementwise;

## 3.2 Objective function in training Neural Network

The variables we want to optimize are the weight matrices, and bias neurons. Then training neural network is to minimize the below objective function:

$$\underset{W_i, b_i}{\text{minimize}} \sum_{j=1}^{M} L(W, b, x_j) \tag{1}$$

Denote $\theta \in R^d$ as the variable of loss function, $\Theta$ as the domain of $\theta$. Then the dimension $d$ of $\theta$ should equal to the sum of dimensions of all weight matrices, and bias terms. Each component of $\theta$ equals to one unique component in one of weight matrices, and bias neurons. Define a mapping $\Phi$, that can map weight matrix, or bias neuron to its corresponding entries in $\theta$, then

$$\Phi(W_1, W_2, \cdots, W_{D-1}, b_1, b_2, \cdots, b_{D-1}) = \theta \tag{2}$$

There are many forms for $L$. For example, if $L$ is quadratic loss, then in MLP:

$$L(\theta, x_j) = ||y_j - softmax(\sigma(\cdots\sigma(W_2\sigma(W_1x_j + b_1) + b_2)) + \cdots + b_D)||_2^2 \tag{3}$$

$$L(\theta) = \sum_{j=1}^{M} ||y_j - softmax(\sigma(\cdots\sigma(W_2\sigma(W_1x_j + b_1) + b_2)) + \cdots + b_D)||_2^2 \tag{4}$$

Problem (1) can be rewritten as

$$\underset{\theta\in\Theta}{\text{minimize}} \sum_{j=1}^{M} L(\theta, x_j) = L(\theta) \tag{5}$$

Without specific constraints, we can set $\Theta = R^{dim(\theta)}$, then $\Theta$ is a convex set. But in terms of regularization, people don't expect the magnitude of $\theta$ too big, we assume $\Theta$ is a **Closed Bounded Set**. However, unfortunately, $L(\theta)$ is often not a convex function for $\theta$.

We also add the below assumptions for the objective function $L(\theta)$:

1. Assume $L(\theta)$ is continuously differentiable on $\Theta$. This assumption makes a lot sense since most of loss function used in neural network, e.g. cross entropy, square loss, are continuously differentiable, unless people select others, like hinge loss.

2. Assume $L(\theta)$ has Lipschitz continuous gradient. This assumption also makes sense since both cross entropy loss and square loss has differentiable continuous gradient. By some results about Lipschitz continuity, continuously differentiable on bounded closed set implies Lipschitz continuous. Then cross entropy and square loss has Lipschitz continuous gradient.

## 3.3 Gradient Calculation of Loss function $L(\theta)$ over $\theta$

In this section, the relationship between backpropagation and $\frac{L(\theta)}{\partial\theta}$, gradient of $L(\theta)$ is shown.

**Lemma 1:** Backpropagation(through time) in Training Scheme 1, 2 produces exact components in the gradient of loss function over $\theta$. Namely if $\frac{\partial L}{\partial W_2}$ is calculated by one backpropagation, then the components of $\frac{\partial L}{\partial W_2}$ are the same as the components in $\frac{\partial L}{\partial \theta}|_{\Phi(W_2)}$.

**Proof:** For any backpropagation in Training Scheme 1, 2, only $\frac{\partial L}{\partial W_i}$ or $\frac{\partial L}{\partial b_i}$ is calculated for $i = 1, 2, \cdots, D-1$.

For $\frac{\partial L}{\partial b_i}$, we have:

$$\frac{\partial L}{\partial b_i} = \begin{pmatrix} \frac{\partial L}{\partial [b_i]_1} \\ \frac{\partial L}{\partial [b_i]_2} \\ \vdots \end{pmatrix}$$

For $\frac{\partial L}{\partial W_i}$, we have:

$$\frac{\partial L}{\partial W_i} = \begin{pmatrix} \frac{\partial L}{\partial [W_i]_{11}} & \frac{\partial L}{\partial [W_i]_{12}} & \cdots \\ \frac{\partial L}{\partial [W_i]_{21}} & \frac{\partial L}{\partial [W_i]_{22}} & \cdots \\ \vdots & \vdots & \vdots \end{pmatrix}$$

Then it is easy to gradient of $L$ either over $W_i$ or $b_i$ is actually gradient of $L$ over each component in $W_i$, and $b_i$. Since $\theta$ shares the same components with $W_i$, and $b_i$, it follows that the components of $\frac{\partial L}{\partial W_i}$ and $\frac{\partial L}{\partial b_i}$ equal to the corresponding entries in $\frac{\partial L}{\partial \theta}$. Based on Lemma 1, we have the following Lemmas:

**Lemma 2:** Backpropagation(through time) in Training Scheme 1 generates full gradient of $L(\theta)$ over $\theta$.

**Proof:** Each time backpropagation in Training Scheme 1 passes through all layers in neural network one time. After calculating $\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2}, \cdots, \frac{\partial L}{\partial W_{D-1}}, \frac{\partial L}{\partial b_1}, \frac{\partial L}{\partial b_2}, \cdots, \frac{\partial L}{\partial b_{D-1}}$, then $\theta$ is updated. By (2), we have

$$\frac{\partial L(\theta)}{\partial \theta} = \Phi(\frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial W_2}, \cdots, \frac{\partial L}{\partial W_{D-1}}, \frac{\partial L}{\partial b_1}, \frac{\partial L}{\partial b_2}, \cdots, \frac{\partial L}{\partial b_{D-1}})$$

that is the backpropagation in Training Scheme 1 can produce full gradient of $L(\theta)$ over $\theta$.

**Lemma 3:** Backpropagation(through time) in Training Scheme 2 generates reduced gradient of $L(\theta)$ over $\theta$.

**Proof:** Each backpropagation in Training Scheme 2 just passes on one layer, and just update variables in current layer. By Lemma 1, backpropagation in Training Scheme 2 only generates reduced gradient of $L(\theta)$ over $\theta$.

## 3.4 Convergence of Training Scheme 1, 2

Based on Section 3.2, 3.3, Training Scheme 1 can be regarded as using constant step size gradient descent method on minimizing nonconvex bounded below continuously differentiable function with Lipschitz continuous gradient. Unfortunately, I can not provide convergence result or some descent result with such weak assumptions. Training Scheme 2 can be regarded as using a constant step size reduced gradient descent method on minimizing nonconvex bounded below continuously differentiable function with Lipschitz continuous gradient. The conditions of Training Scheme 2 is even weaker than Training Scheme 1. Of course, no convergence result can be offered. People can only empirically choose a sufficiently small learning rate ( step size ) to see whether the loss function can be reduced or not.

## 3.5 Training Scheme 3

Even though the convergence of Training Scheme 1,2 are disappointed, we can add back-tracking like strategy into Training Scheme 1, shown in below:

**Algorithm 3** Training Scheme 3

```
1:  procedure INPUT DATASET $X$
2:      Initialize weights, bias parameter. Select $c_1, c_2$ such that $0 < c_1 < c_2 < 1, 0 < \xi < 1$.
3:  loop for $k = 0, 1, 2, 3, \cdots$:
4:      For each $x \in X$:
5:          Forward propagate to get predicted output $\hat{y}$ and calculate loss error
6:          Back propagate(through time) to calculate gradients for each variable
7:      Calculate $||\nabla L(\theta_k)||_2^2$ by summing squared elements in gradient of each variable
8:      Calculate Trial Points $\theta_{trial}$ by
            $$\theta_{trial} = \theta_k - lr_k \cdot \nabla L(\theta_k)$$
9:      Forward Propagate $X$ to calculate new loss $L(\theta_{trial})$ based on $\theta_{trial}$
10:     if $L(\theta_{trial}) \leq L(\theta_k) - c_1 \cdot lr_k \cdot ||\nabla L(\theta_k)||_2^2$ and $\nabla L^T(\theta_{trial}) \nabla L(\theta_k) \leq c_2 ||\nabla L(\theta_k)||_2^2$ then
11:         goto Line 16;
12:     else
13:         $lr_k \leftarrow \xi \cdot lr_k$
14:         goto Line 9.
15:     Set $\theta_{k+1} \leftarrow \theta_{trial}$
16:     if $k >=$ maximum training epoches then Stop;
17:     else
18:         goto Line 4.
```

Line 11 is Wolfe condition with steepest descent direction. $\frac{\partial L(\theta_{trial})}{\partial \theta_{trial}}$ can be obtained by one more backpropagation. Based on the assumptions in previous sections, we have the following Theorem.

***Theorem 1*** *If loss function $L(\theta)$ is continuously differentiable, bounded below on level set $\{\theta : L(\theta) \leq L(\theta_0)\}$ and has Lipschitz continuous gradient, then*

$$\lim_{k\to\infty} ||\nabla L(\theta_k)|| = 0$$

**Proof:** By Line 11 to Line 16 in Training Scheme 3, we have

$$\left(\nabla L(\theta_{k+1}) - \nabla L(\theta_k)\right)^T \left(-\nabla L(\theta_k)\right) \geq (1 - c_2)\nabla^T L(\theta_k)\nabla L(\theta_k) \tag{6}$$

Lipschtiz Continuity of gradient of $L(\theta)$ implies there exists a positive constant $L_g$, such that

$$\left(\nabla L(\theta_{k+1}) - \nabla L(\theta_k)\right)^T \left(-\nabla L(\theta_k)\right) \leq lr_k \cdot L_g \cdot \nabla^T L(\theta_k)\nabla L(\theta_k) \tag{7}$$

By (6), (7), we have

$$(1 - c_2)\nabla^T L(\theta_k)\nabla L(\theta_k) \leq lr_k \cdot L_g \cdot \nabla^T L(\theta_k)\nabla L(\theta_k)$$

$$lr_k \geq \frac{1 - c_2}{L_g} > 0 \tag{8}$$

It follows that

$$L(\theta_{k+1}) \leq L(\theta_k) - c_1\frac{1 - c_2}{L_g}||\nabla L(\theta_k)||_2^2 \tag{9}$$

Summing Inequality (9) from $k = 0$ to any $T \geq 1$, we have

$$\sum_{k=1}^{T} L(\theta_k) \leq \sum_{k=0}^{T} L(\theta_k) - \frac{c_1(1 - c_2)}{L_g}\sum_{k=0}^{T}||\nabla L(\theta_k)||_2^2 \tag{10}$$

Substract $\sum_{k=1}^{T-1} L(\theta_k)$ on both sides of (10), then

$$L(\theta_T) \leq L(\theta_0) - \frac{c_1(1 - c_2)}{L_g}\sum_{k=0}^{T}||\nabla L(\theta_k)||_2^2 \tag{11}$$

Since $L(\theta)$ is bounded below on level set $\{\theta : L(\theta) \leq L(\theta_0)\}$, it follows that

$$\frac{c_1(1-c_2)}{L_g} \sum_{k=0}^{T} ||\nabla L(\theta_k)||_2^2 < \infty \tag{12}$$

By $0 < c_1 < c_2 < 1$, we have $\frac{c_1(1-c_2)}{L_g} > 0$, consequently,

$$\sum_{k=0}^{T} ||\nabla L(\theta_k)||_2^2 < \infty \tag{13}$$

Since (13) holds for any $T$, then

$$\sum_{k=0}^{\infty} ||\nabla L(\theta_k)||_2^2 < \infty \tag{14}$$

By (14), we can conclude that

$$\lim_{k \to \infty} ||\nabla L(\theta_k)|| = 0$$

Based on Theorem 1, we have the following Corollary:

***Corollary 1*** *Training Scheme 3 converges to a stationary point $\theta_k$ as $k$ tends to $\infty$.*

## 3.6 Training Scheme 4 with noised gradient

Now, **assume we can get real loss function, but not exact gradient**. That is at training epoch $k$, the gradient $\nabla \hat{L}(\theta_k)$ obtained from back propagation is

$$\nabla \hat{L}(\theta_k) = \nabla L(\theta_k) + e(\theta_k) \tag{15}$$

where $\nabla L(\theta_k)$ is real gradient, $e(\theta_k)$ is noise. Without real gradient, we can not guarantee there exists an interval such that Wolfe Condition holds. Hence, Training Scheme 3 is modified into Training Scheme 4.

**Algorithm 4** Training Scheme 4

```
1: procedure INPUT DATASET $X$
2:     Initialize weights, bias parameter. Select $c_1, c_2$ such that $0 < c_1 < c_2 < 1, 0 < \xi < 1$.
3: loop for $k = 0, 1, 2, 3, \cdots$:
4:     For each $x \in X$:
5:         Forward propagate to get predicted output $\hat{y}$
6:         Calculate loss error
7:         Back propagate(through time) to calculate gradients for each variable
8:     Calculate Trial Points $\theta_{trial}$ by
           $$\theta_{trial} = \theta_k - lr_k \cdot \nabla \hat{L}(\theta_k)$$
9:     Forward Propagate $X$ to calculate new loss $L(\theta_{trial})$ based on $\theta_{trial}$
10:    if $L(\theta_{trial}) \leq L(\theta_k)$ then
11:        goto Line 15;
12:    else
13:        $lr_k \leftarrow \xi \cdot lr_k$
14:        goto Line 8.
15:    Set $\theta_{k+1} \leftarrow \theta_{trial}$
16:    if $k >=$ maximum training epoches then
17:        Stop;
18:    else
19:        goto Line 4.
```

Next define the following events:

$$DD_k = \{\ cos(\phi_k) := \frac{\nabla \hat{L}^T(\theta_k)\nabla L(\theta_k)}{||\nabla \hat{L}(\theta_k)||_2^2} \geq \delta > 0\ \}$$

$$SD_k = \{\ L(\theta_{k+1}) \leq L(\theta_k) - c_1 \cdot lr_k \nabla \hat{L}^T(\theta_k)\nabla L(\theta_k)\ \}$$

$$CC_k = \{\ \nabla L^T(\theta_{k+1})\nabla \hat{L}(\theta_k) \leq c_2 \nabla \hat{L}^T(\theta_k)\nabla L(\theta_k)\ \}$$

for some constant $0 < \delta, c_1, c_2 < 1$. Let $\mathcal{F}_k$ denote the $\sigma$-algebra generated by

$$DD_0, SD_0, CC_0, DD_1, SD_1, CC_1, \cdots, DD_{k-1}, SD_{k-1}, CC_{k-1}$$

Then we have the following Theorem:

***Theorem 2*** *If $L(\theta)$ is continuously differentiable, bounded below on level set and has Lipschitz continuous gradient. If there exists three constant $0 < \gamma \leq 1$, $0 < \alpha \leq 1$, $0 < \beta \leq 1$, such that*

$$Pr(DD_k|\mathcal{F}_k) \geq \alpha \tag{16}$$

$$Pr(SD_k|\mathcal{F}_k) \geq \beta \tag{17}$$

$$Pr(CC_k|\mathcal{F}_k) \geq \gamma \tag{18}$$

*for any $k = 0, 1, 2, \cdots$, assume $DD_0, SD_0, CC_0, SD_1, CV_1, \cdots, DD_{k-1}, SD_{k-1}, CC_{k-1}$ are i.i.d., then,*

$$\lim_{k \to \infty} ||\nabla L(\theta_k)||_2^2 = 0 \tag{19}$$

**Proof:** At training epoch $k$, there exist two cases:

**Case 1:** All $DD_k$, $SD_k$ ,and $CC_k$ occur:

Then, by the proof of Theorem 1, it follows that

$$\begin{aligned} L(\theta_{k+1}) - L(\theta_k) &\leq -\frac{c_1(1-c_2)}{L_g} \frac{\left[\nabla L^T(\theta_k)\nabla \hat{L}(\theta_k)\right]^2}{||\nabla \hat{L}(\theta_k)||_2^2} \\ &= -\frac{c_1(1-c_2)}{L_g} (cos\phi_k)^2 \, ||\nabla L(\theta_k)||_2^2 \\ &\leq -\frac{c_1(1-c_2)}{L_g} \delta^2 ||\nabla L(\theta_k)||_2^2 \end{aligned} \tag{20}$$

By the assumption that $DD_0, SD_0, CC_0, SD_1, CV_1, \cdots, DD_{k-1}, SD_{k-1}, CC_{k-1}$ are *i.i.d.*, then we have

$$Pr(A_k) := Pr(DD_k \cap SD_k \cap CC_k) \geq \alpha \cdot \beta \cdot \gamma \tag{21}$$

**Case 2:** At least one of $DD_k$, $SD_k$ ,and $CC_k$ is not statisfied:

By Line 10, we have

$$L(\theta_{k+1}) - L(\theta_k) \leq 0 \tag{22}$$

By the assumptions,

$$Pr(B_k) := Pr((DD_k \cap SD_k \cap CC_k)^C) \leq 1 - \alpha \cdot \beta \cdot \gamma \tag{23}$$

Based on Case 1, 2, it follows that

$$\begin{aligned} E(L(\theta_{k+1}) - L(\theta_k)) &< -\frac{c_1(1-c_2)}{L_g} \delta^2 ||\nabla L(\theta_k)||_2^2 \cdot Pr(A_k) + 0 \cdot Pr(B_k) \\ &\leq -\frac{c_1(1-c_2)}{L_g} \delta^2 ||\nabla L(\theta_k)||_2^2 \cdot \alpha \cdot \beta \cdot \gamma \\ &= -c||\nabla L(\theta_k)||_2^2 \end{aligned} \tag{24}$$

where let $c$ denote constant: $\frac{c_1(1-c_2)}{L_g}\delta^2 \cdot \alpha \cdot \beta \cdot \gamma$, notice that $c > 0$.

Summing inequalities (24) from $k = 0$ to T, we have

$$E(L(\theta_{T+1}) - L(\theta_0)) \leq -c \sum_{k=0}^{T} ||\nabla L(\theta_k)||_2^2 \tag{25}$$

$$E(L(\theta_{T+1})) \leq E(L(\theta_0)) - c \sum_{k=0}^{T} ||\nabla L(\theta_k)||_2^2 \tag{26}$$

By the assumption that $L(\theta)$ is bounded below on the level set, it follows that $E(L(\theta_{T+1})) > -\infty$ . Consequently, with $c > 0$, then

$$\sum_{k=0}^{T} ||\nabla L(\theta_k)||_2^2 < \infty \tag{27}$$

Since (27) holds for any $T$, then

$$\sum_{k=0}^{\infty} ||\nabla L(\theta_k)||_2^2 < \infty$$

It follows that

$$\lim_{k\to\infty} ||\nabla L(\theta_k)||_2^2 = 0$$

Based on Theorem 2, we can obtain the the following Corollary:

***Corollary 2*** *Under the assumption in Theorem 2, $\theta$ produced by Training Scheme 4 converges to a stationary point of $L(\theta)$ on $\Theta$.*

Theorem 2, and Corollary 2 claim the convergence of Training Scheme 4 to stationary point with $Pr(DD_k|\mathcal{F}_k), Pr(SD_k|\mathcal{F}_k), Pr(CC_k|\mathcal{F}_k)$ are bounded away 0, namely there are positive constants $0 < \alpha, \beta, \gamma \leq 1$, such that

$$Pr(DD_k|\mathcal{F}_k) \geq \alpha$$

$$Pr(SD_k|\mathcal{F}_k) \geq \beta$$

$$Pr(CC_k|\mathcal{F}_k) \geq \gamma$$

For weaker condition, we have the following Theorem 3.

***Theorem 3*** *If $L(\theta)$ is continuously differentiable, bounded below on level set and has Lipschitz continuous gradient. If*

$$Pr(DD_k|\mathcal{F}_k) \geq 0 \tag{28}$$

$$Pr(SD_k|\mathcal{F}_k) \geq 0 \tag{29}$$

$$Pr(CC_k|\mathcal{F}_k) \geq 0 \tag{30}$$

*for any* $k = 0, 1, 2, \cdots,$ *and*

$$\sum_{k=0}^{\infty} Pr(DD_k \cap SD_k \cap CC_k) = \infty \tag{31}$$

*then Training Scheme 4 generate* $\theta_k$ *such that*

$$\underset{k \to \infty}{lim\ inf} ||\nabla L(\theta_k)||_2^2 = 0 \tag{32}$$

**Proof:** Proof on contradiction.

Suppose there exists a constant $\epsilon > 0$, such that

$$\liminf_{k \to \infty} \ ||\nabla L(\theta_k)||_2^2 \geq \epsilon > 0 \tag{33}$$

At training epoch $k$, there exist two cases:

**Case 1:** All $DD_k, SD_k, CC_k$ occur:

By the proof of Theorem 2, we have

$$L(\theta_{k+1}) - L(\theta_k) \leq -c \cdot ||\nabla L(\theta_k)||_2^2 \tag{34}$$

where $c = \frac{c_1(1-c_2)}{L_g}\delta^2$.

**Case 2:** At least one of $DD_k, SD_k, CC_k$ does not hold:

By Line 10 in Training Scheme 4,

$$L(\theta_{k+1}) - L(\theta_k) \leq 0 \tag{35}$$

By (33), there exists a $K > 0$, such for any $k \geq K$,

$$||\nabla L(\theta_k)||_2^2 \geq \epsilon$$

Then, for any $k \geq K$, it follows that

$$\begin{aligned} E(L(\theta_{k+1}) - L(\theta_k)) &\leq -c \cdot \epsilon \cdot Pr(DD_k \cap SD_k \cap CC_k) \\ &\quad + 0 \cdot Pr((DD_k \cup SD_k \cup CC_k)^C) \\ &= -c \cdot \epsilon \cdot Pr(DD_k \cap SD_k \cap CC_k) \end{aligned} \tag{36}$$

Summing the above inequality from $k = K$ to $k = T$, then

$$E(L(\theta_{T+1}) - L(\theta_K)) \leq -c \cdot \epsilon \cdot \sum_{k=K}^{T} Pr(DD_k \cap SD_k \cap CC_k) \tag{37}$$

$$E(L(\theta_{T+1}) \leq E(L(\theta_K)) - c \cdot \epsilon \cdot \sum_{k=K}^{T} Pr(DD_k \cap SD_k \cap CC_k) \tag{38}$$

By bounded below assumption, it follows that

$$\sum_{k=K}^{\infty} Pr(DD_k \cap SD_k \cap CC_k) < \infty \tag{39}$$

But since $0 \leq Pr(DD_k \cap SD_k \cap CC_k) \leq 1$ for any $k$, and (31) holds, then we have

$$\sum_{k=K}^{\infty} Pr(DD_k \cap SD_k \cap CC_k) = \infty \tag{40}$$

which contradicts with (39). Therefore,

$$\liminf_{k \to \infty} \; ||\nabla L(\theta_k)||_2^2 = 0$$

## 3.7 Decaying Learning Rate Sequence Training Scheme 5

Finally, let's consider that decaying learning rate strategy with noised back-propagation, shown in Training Scheme 5.

**Algorithm 5** Training Scheme 5

```
1: procedure INPUT DATASET $X$
2:     Initialize weights, bias parameter. Initialize $lr_0 > 0$.
3: loop for $k = 0, 1, 2, 3, \cdots$:
4:     For each $x \in X$:
5:         Forward propagate to get predicted output $\hat{y}$
6:         Calculate loss error
7:         Back propagate to calculate gradients for each variable in each layer
8:     Update $\theta$ by
               $\theta_{k+1} = \theta_k - lr_k \cdot \nabla \hat{L}(\theta_k)$
9:     Update $lr_{k+1}$
```

The sequence of learning rate $lr_k$ should satisfy the following conditions:

$$lr_k > 0,\ \sum_{k=0}^{\infty} lr_k = \infty,\ \sum_{k=0}^{\infty} lr_k^2 < \infty$$

We have the following theorem:

***Theorem 4*** *If $||\nabla \hat{L}(\theta_k)||_2$ is bounded by constant $B > 0$, $E(\nabla \hat{L}^T(\theta_k)(\theta_k - \theta^*)|\mathcal{E}^k) \geq 0$, where $\theta^*$ is one global minimizer, $\mathcal{E}^k$ is the $\sigma$-algebra of previous conditions. Then Training Scheme 5 generate a sequence $\{\theta_k\}$, such that*

$$\lim_{k \to \infty} ||E(\nabla L(\theta_k))||_2^2 = 0 \tag{41}$$

*almost surely.*

**Proof:** Since most of proof is the same as [8] and maximum page is not more than 18, here I just point out the differences. At first, Assumption 3 (A3) in [8] is removed:

*A3 : The expected gradient cannot vanish at a non-optimal value of $\theta$. Therefore, for any $\theta$ such that $|\theta - \theta^*| > 0$, there exists constant $\epsilon > 0$ such that $E(||\nabla \hat{L}(\theta)||) \geq \epsilon$.*

A3 is removed since $L(\theta)$ is not convex, A3 is rare to be satisfied.

For the proof in last slide of [8], the third scenario should be removed since A3 does not hold. Consequently, there exist two cases totally:

**Case 1:** $\theta_k$ converges to $\theta^*$as $k \to \infty$ almost surely .

**Case 2:** $E(\nabla \hat{L}(\theta_k))$ converges to $\vec{0}$ as $k \to \infty$ almost surely.

Both cases implies that

$$||E(\nabla L(\theta_k))||_2^2 \to 0$$

almost surely as $k \to \infty$. Therefore,

$$\lim_{k \to \infty} ||E(\nabla L(\theta_k))||_2^2 = 0$$

almost surely.

Based on Theorem 4, we have Corollary 3:

***Corollary 3*** *With the assumptions in Theorem 4, Training Scheme 5 can generate a sequence $\{\theta_k\}$, such that $\theta_k$ converges a stationary point in expectation almost surely as $k \to \infty$.*

## 4 Recurrent neural network on language model

This section is added in this report because in my project proposal, language model on Recurrent Neural Network is mentioned. However since I have used this section as my another course project this semester: EN.600.468 - Machine Translation, instead of describing it in this report, I just attach the report for EN.600.468 in the submitted project folder. In that report, two RNNs: LSTM and GRU are derived, and implemented in Training Scheme 1 ( source code can be found at my github repository [7]). The numerical experiments show GRU can reach competitive BLEU score comparing with the n-gram counting language model provided by the instructor of EN 600.468: Prof. Phillip Koehn.

## 5 Conclusion

So far, two constant learning rate training schemes can not guarantee convergence. Backtracking training scheme, backtracking noised training scheme, and decaying learning rate noised training scheme can converge to stationary point in different forms.

## References

[1] McCulloch, Warren; Walter Pitts. *A Logical Calculus of Ideas Immanent in Nervous Activity.* Bulletin of Mathematical Biophysics 5 (4): 115?133. 1943

[2] Rosenblatt, F. *The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain.* Psychological Review 65 (6): 386?408. doi:10.1037/h0042519. PMID 13602029. 1958

[3] Werbos, P.J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* 1975

[4] R. Collobert and S. Bengio. *Links between Perceptrons, MLPs and SVMs.* Proc. Int'l Conf. on Machine Learning (ICML). 2004

[5] https://github.com/motian12ps/DLcty/blob/master/RBM

[6] Y. Benigio, P. Simard, P. Frasconi, *Learning Long-Term Dependence with Gradient Descent is Difficult*, IEEE Transaction on Neural Network, Vol 5, No. 2, 1994

[7] https://github.com/motian12ps/LSTM-GRU

[8] Warren Powell, *Stochastic Optimization Seminar Lecture 2:The Stochastic Gradient Algorithm and Proof of Convergence*, 2012